SIAM REVIEW
Vol. 45, No. 3, pp. 385–482

© 2003 Society for Industrial and Applied Mathematics

Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods*

Tamara G. Kolda†
Robert Michael Lewis‡
Virginia Torczon§

Abstract. Direct search methods are best known as unconstrained optimization techniques that do not explicitly use derivatives. Direct search methods were formally proposed and widely applied in the 1960s but fell out of favor with the mathematical optimization community by the early 1970s because they lacked coherent mathematical analysis. Nonetheless, users remained loyal to these methods, most of which were easy to program, some of which were reliable. In the past fifteen years, these methods have seen a revival due, in part, to the appearance of mathematical analysis, as well as to interest in parallel and distributed computing.

This review begins by briefly summarizing the history of direct search methods and considering the special properties of problems for which they are well suited. Our focus then turns to a broad class of methods for which we provide a unifying framework that lends itself to a variety of convergence results. The underlying principles allow generalization to handle bound constraints and linear constraints. We also discuss extensions to problems with nonlinear constraints.

Key words. nonlinear programming, nonlinear optimization, direct search, pattern search, simplex search, positive bases, global convergence analysis, local convergence analysis, generating set search

AMS subject classifications. 90C56, 90C30, 65K05

DOI. 10.1137/S0036144502428893

Contents.

1 Introduction **387**
1.1 An Early, Simple Direct Search Method 389
1.2 Applicability of Direct Search 390

*Received by the editors December 20, 2002; accepted for publication (in revised form) July 8, 2003; published electronically August 11, 2003. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.
http://www.siam.org/journals/sirev/45-3/42889.html

†Computational Sciences and Mathematics Research Department, Sandia National Laboratories, Livermore, CA 94551-9217 (tgkolda@sandia.gov). The work of this author was supported by the Mathematical, Information, and Computational Sciences Program of the U.S. Department of Energy under contract DE-AC04-94AL85000 with Sandia Corporation.

‡Department of Mathematics, College of William & Mary, P.O. Box 8795, Williamsburg, VA 23187-8795 (buckaroo@math.wm.edu). The work of this author was supported by the Computer Science Research Institute at Sandia National Laboratories, by the National Science Foundation under grant DMS-0215444, and by the National Aeronautics and Space Administration under grant NCC-1-02029.

§Department of Computer Science, College of William & Mary, P.O. Box 8795, Williamsburg, VA 23187-8795 (va@cs.wm.edu). The work of this author was funded by the Computer Science Research Institute at Sandia National Laboratories and by the National Science Foundation under grant CCR-9734044.

1.2.1 Simulation-Based Optimization . . . 391
1.2.2 Nonnumerical Functions . . . 392
1.3 Issues When Using Direct Search Methods . . . 393
1.3.1 Slow Asymptotic Convergence . . . 393
1.3.2 Limitations on Problem Size . . . 393
1.4 What's in a Name? . . . 394
1.5 Objectives of This Review . . . 395
1.6 Topics Outside the Scope of This Review . . . 395

2 Smooth Unconstrained Minimization 396
2.1 Global Convergence and Local Convergence . . . 396
2.2 Line Search Methods . . . 397
2.3 Avoiding Poor Choices of Step Lengths . . . 398
2.4 Avoiding Poor Choices of Descent Directions . . . 399
2.5 A Global Convergence Result for Line Search Methods . . . 399

3 Generating Set Search Methods for Unconstrained Minimization 400
3.1 An Example: Compass Search Revisited . . . 401
3.2 Convergence of Compass Search . . . 402
3.3 Generating Set Search . . . 404
3.3.1 Bounds on the Lengths of the Directions . . . 406
3.3.2 Update Formulas . . . 406
3.3.3 Step Acceptance Criteria . . . 406
3.4 Generating Sets for $\mathbb{R}^n$ . . . 406
3.4.1 Generating Sets and Descent . . . 407
3.5 The Ingredients for Proving Global Convergence . . . 409
3.6 Relating Δ_k to the Measure of Stationarity . . . 409
3.7 Globalization Strategies in GSS . . . 410
3.7.1 Globalization via the Sufficient Decrease Condition . . . 411
3.7.2 Globalization via a Rational Lattice . . . 413
3.7.3 Globalization via Moving Grids . . . 419
3.8 Putting It All Together—Global Convergence Results . . . 421
3.8.1 The General Result . . . 421
3.8.2 A Stronger Result . . . 422
3.9 Some Comments on the Convergence of GSS under Simple Decrease . 424
3.10 Stopping Criteria . . . 424
3.11 Local Convergence of GSS . . . 425

4 Algorithmic Options 427
4.1 Statistical Design of Experiments . . . 428
4.2 Exploratory Moves . . . 431
4.3 Using Oracles . . . 433

5 Variations on the GSS Requirements 435
5.1 Controlling the Step Length . . . 436
5.2 Additional Algorithmic Steps . . . 437
5.3 Independent Directions for Parallel Computing . . . 437

6 The Role of Smoothness in the Robustness of GSS Algorithms 437
6.1 The Effects of Multiple Search Directions and Sampling . . . 438

6.2 Why We Assume f Is Continuous 440
6.3 Why We Assume ∇f Exists and Is Continuous 440
6.4 GSS Methods and Lipschitz Optimization 441
6.5 The Gap between Theory and Practice 443

7 Some Background on Constrained Optimization 443
7.1 The Effects of Constraints 444
7.2 The Geometry of Cones 445
7.3 First-Order Necessary Conditions for a Constrained Minimizer 447
7.4 Measures of Convergence to KKT Points 448

8 Direct Search and Constrained Optimization 449
8.1 The Case of Bound Constraints: Compass Search, Again 450
8.1.1 The Geometry of the Nearby Boundary 451
8.1.2 Obtaining Feasible Descent 452
8.1.3 A Bound on Stationarity at Unsuccessful Iterations 455
8.2 Direct Search Methods for General Linear Constraints 456
8.2.1 The Geometry of the Nearby Boundary 457
8.2.2 Specifying the Generating Set 457
8.2.3 GSS Algorithms for Linearly Constrained Problems 459
8.2.4 Existence of a Feasible Descent Direction among the Search Directions 460
8.2.5 The Situation at Unsuccessful Iterations 464
8.2.6 Global Convergence Results 465
8.2.7 Stopping Criteria 466
8.2.8 Constructing the Search Directions 466
8.3 General Nonlinear Constraints 466
8.3.1 Feasible Direction Methods That Use Constraint Derivatives . 467
8.3.2 An Augmented Lagrangian Approach 468
8.3.3 Other Approaches 469

9 Summary 470

1. Introduction. Consider the problem of finding a local minimizer of a real-valued function $f(x)$. If f is differentiable and $\nabla f(x)$ can be computed or accurately estimated by finite-differences, a wide selection of derivative-based optimization methods is available to solve this problem. Now suppose that explicit information about $\nabla f(x)$ is unavailable or untrustworthy. Can this minimization problem still be solved? The task of minimizing a function without recourse to its derivatives may seem impossible since, for a differentiable function, a necessary condition for a minimizer is that the gradient vanishes. Can such a point be found without explicitly using derivatives? *Direct search methods* attempt to do just this. An appreciable number of them succeed.

Direct search methods have been known since at least the 1950s (see [171] for a historical review). However, by the early 1970s, these methods were largely dismissed by the mathematical optimization community and disappeared from most of its literature for three basic reasons summarized in Swann's 1971 comments [250]:

> . . . [direct search] methods have been developed heuristically, . . . no proofs of convergence have been derived for them, . . . [and] sometimes the rate of convergence can be very slow.

Nevertheless, direct search methods remained popular with practitioners in the scientific and engineering communities. Early direct search methods were based on simple and appealing heuristics often derived from examples drawn in two dimensions (see, for example, the development in [227]). In addition, many users preferred to avoid the calculation of gradients, which was for a long time the single biggest source of error in applying optimization software libraries [117, p. 297].

In 1991, interest in direct search methods was revived with the publication, in the context of parallel computing, of a direct search method with an accompanying convergence analysis [256]. Since then, two things have become increasingly clear.

1. Direct search methods remain an effective option, and sometimes the only option, for several varieties of difficult optimization problems.
2. For a large number of direct search methods, it is possible to provide rigorous guarantees of convergence.

Both of these points will be considered in this review, emphasizing the second in great detail. Ironically, many of the key elements necessary to prove convergence already had been identified [21, 22, 56, 211] by the time these algorithms were exiled by the mathematical optimization community.

The lack of a precise definition of a "direct search method" adds complications. We attempt to clarify the definition in section 1.4. The term appears to have originated in the 1961 paper by Hooke and Jeeves [139] but since then has become a catch-all phrase that often is applied to any optimization method that does not require an explicit representation of the gradient of f. Rather than attempt a complete survey of what has become a large and disparate class of approaches, we focus on a particular subclass of direct search methods. Our theme will be the key algorithmic features that unify this subclass and lead to a variety of convergence results. This subclass includes the pattern search method first proposed in the direct search paper by Hooke and Jeeves [139], as well as a variety of more recent developments.

In section 4.1 we briefly discuss the *Nelder–Mead simplex algorithm* [194], probably the most widely cited of the direct search methods (the 1965 paper by Nelder and Mead is officially a *Science Citation Classic*), but we refer interested readers to [164] for a comprehensive modern treatment of the original Nelder–Mead simplex algorithm. In the same section we briefly mention the *derivative-free methods* that rely on local models of the objective and constraints. These methods have received much attention of late and show some connections with the ideas presented here, but for a more thorough treatment we refer interested readers to the reviews [72, 221]. A related topic that is omitted is the study of unidirectional line searches that do not require derivatives, such as those in [42, 182, 183]. We mention in passing the use of purposefully coarse finite-difference derivatives, an idea first developed in [178, 181] and appearing more recently under the name *implicit filtering* (see [154] for details). Finally, since not all the algorithms that now fall under the general term "direct search methods" fit neatly into any one taxonomy, we have included extensive references (not all of which are discussed within the text) pointing to additional work that may be of interest.

For our part, we will stick most closely to the spirit of Hooke and Jeeves' original definition of direct search (see section 1.4). To illustrate what we mean by direct search, and to show the appealing simplicity of early approaches, in the next subsection we present the most straightforward, and probably the best known, direct search method.

1.1. An Early, Simple Direct Search Method. In his belated preface [83] for ANL 5990 [82], Davidon describes what is one of the earliest examples of a direct search method used on a digital computer to solve an optimization problem:

> Enrico Fermi and Nicholas Metropolis used one of the first digital computers, the Los Alamos Maniac, to determine which values of certain theoretical parameters (phase shifts) best fit experimental data (scattering cross sections) [105]. They varied one theoretical parameter at a time by steps of the same magnitude, and when no such increase or decrease in any one parameter further improved the fit to the experimental data, they halved the step size and repeated the process until the steps were deemed sufficiently small. Their simple procedure was slow but sure, and several of us used it on the Avidac computer at the Argonne National Laboratory for adjusting six theoretical parameters to fit the pion-proton scattering data we had gathered using the University of Chicago synchrocyclotron [7].

While this basic algorithm undoubtedly predates Fermi and Metropolis, it has remained a standard in the scientific computing community for exactly the reason observed by Davidon: it is slow but sure.

Davidon's description can be made more concrete with a simple illustration of the algorithm, which we call *compass search*. Variations of this basic algorithmic strategy can be found under numerous names, including *alternating directions*, *alternating variable search*, *axial relaxation*, *coordinate search*, and *local variation* [257]. For a minimization problem with only two variables, the algorithm can be summarized as follows: Try steps to the East, West, North, and South. If one of these steps yields a reduction in the function, the improved point becomes the new iterate. If none of these steps yields improvement, try again with steps half as long.

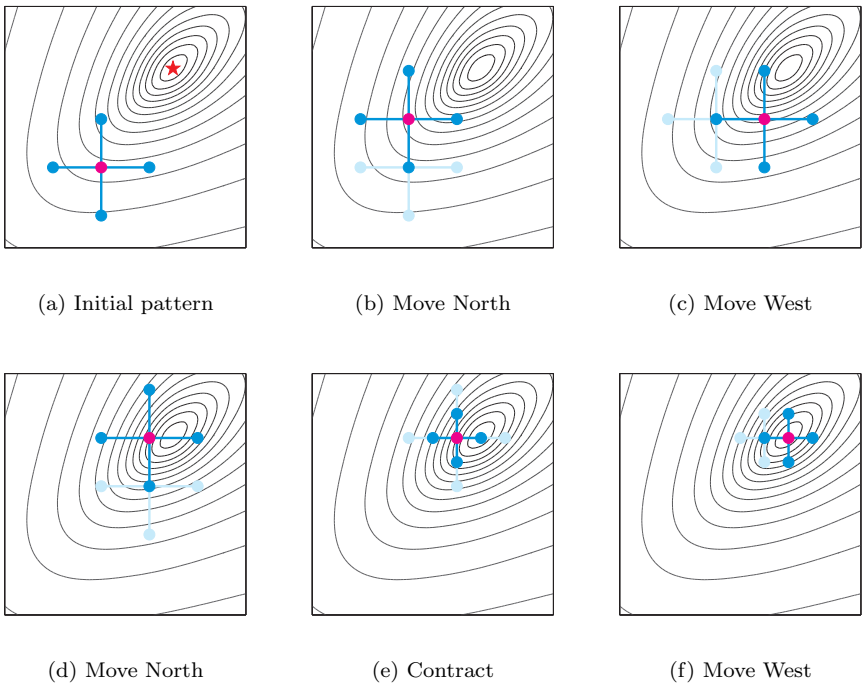
Figure 1.1 illustrates the first five iterations of compass search applied to the problem

$$\underset{x\in\mathbb{R}^2}{\text{minimize}}\ f(x^{(1)},x^{(2)}) = \left|(3-2x^{(1)})x^{(1)} - 2x^{(2)} + 1\right|^{\frac{7}{3}} + \left|(3-2x^{(2)})x^{(2)} - x^{(1)} + 1\right|^{\frac{7}{3}}.$$

The function f is the two-dimensional modified Broyden tridiagonal function [48, 185]. Level curves of f are shown in the background using shades of gray. Here $x^{(1)}$ and $x^{(2)}$ refer to the first and second components of $x \in \mathbb{R}^2$; henceforth, x_k denotes the point that is the current iterate. In each figure, a magenta dot denotes x_k. The current iterate is the best point, i.e., the one with the lowest value of f found so far. In subfigure (a), the search commences from the initial iterate $x_0 = (-0.9, -1.0)$ and the solution to the problem is marked with a red star. Each of the six subfigures represents one iteration of compass search. The four dark blue dots represent the trial points under consideration at that iteration. Taking a step of a given length along each of the four search directions yields the trial points. Initially, the length of each step is 0.3. The trial points from the previous iteration are shown in light blue for comparison.

Note that as x_k approaches the solution, the algorithm reduces the length of the steps taken. This turns out to be central to the convergence proofs. Typically, in practice, the algorithm is terminated when the step length falls below a certain tolerance.

Already the appeal of these methods, and some potential limitations, are clear. On the positive side, compass search is easy to describe and easy to implement. Also, in this example, compass search initially makes rapid progress toward the solution. On the negative side, the fact that the iterates are near a minimizer will only become



(a) Initial pattern (b) Move North (c) Move West

(d) Move North (e) Contract (f) Move West

Fig. 1.1 *Compass search applied to the modified Broyden tridiagonal function.*

apparent as the algorithm reduces the length of the trial steps. That is, the algorithm may quickly approach a minimizer, but may be slow to detect this fact. This is the price of not explicitly using gradient information. Another limitation that is not evident from this example is that the algorithm may be slow to converge if the level sets of f are extremely elongated. This is because the method makes no use of curvature (i.e., second derivative) information. The search may make good progress initially, but in the absence of higher-order information, the asymptotic rate of convergence will be slow.

1.2. Applicability of Direct Search. As mentioned earlier, direct search methods were popular in the early years of numerical optimization in large part because they are straightforward to implement and do not require derivatives. Neither of these reasons is necessarily compelling today. Sophisticated implementations of derivative-based methods, with line search or trust region globalization strategies and options to generate approximations to the gradient (the vector of first partial derivatives) and/or the Hessian (the matrix of second partial derivatives), are widely available and relatively easy to use. Furthermore, today there are automatic differentiation tools ([25, 26, 27, 28, 126]; see also section 1.2.1) as well as modeling languages [45, 111] that compute derivatives automatically. Thus, a user only needs to provide a procedure that calculates the function values. Today, most people's first recommendation (including ours) to solve an unconstrained problem for which accurate first derivatives can be obtained would not be a direct search method, but rather a gradient-based method. If second derivatives were also available, the top choice would be a Newton-based method. (For more details, see [69, 192, 197].)

But this does not mean that direct search methods are no longer needed. They still have their niche. In particular, the maturation of simulation-based optimization has led to optimization problems with features that make it difficult to apply methods that require derivative information. There are also optimization problems where methods based on derivatives cannot be used because the objective function being optimized is not numerical in nature.

1.2.1. Simulation-Based Optimization. The term *simulation-based optimization* is currently applied to the methodology in which complex physical systems are designed, analyzed, and controlled by optimizing the results of computer simulations. For instance, optimization might be used to determine the worst-case radiation exposure in the interior of a space vehicle subjected to solar winds, to control the heat settings for chemical vapor deposition in a hot-wall tubular reactor, or to estimate the values of physical parameters, as in the examples described by Davidon.

In the simulation-based optimization setting, a computer simulation must be run, repeatedly, in order to compute the various quantities needed by the optimization algorithm. Each simulation may in turn involve executing several independent programs in sequence, such as a geometry generator, a mesh generator, and a partial differential equations solver. Furthermore, the resulting simulation output must then be post-processed to arrive finally at values of the objective and constraint functions. These complications can make obtaining derivatives for gradient-based methods at the very least difficult, even when the underlying objective and constraint functions are smooth (i.e., continuously differentiable).

For complex simulations such as those just described, the investment to obtain analytical derivatives typically is substantial. Though automatic differentiation tools have been shown to be quite effective for some types of simulation-based problems, they are not always applicable or appropriate, as discussed by Hovland [143]. For example, although possible in theory, automatic differentiation tools cannot yet be applied to codes that mix C, C++, and Fortran procedures, a common feature in simulations. In addition, automatic differentiation currently cannot be applied when the source code is unavailable, which can be the case when using proprietary or commercial simulation codes.

A standard option, when the function to be optimized is smooth and its calculated values have full precision, is to use finite-differences (with a small interval) to obtain derivative estimates that are accurate enough to be treated as exact gradients in a quasi-Newton method [117]. This brings us to the problems of noise and nonsmoothness.

The term *nonsmooth optimization* typically is used in connection with functions like $|x|$ or the largest eigenvalue of a symmetric matrix, which are Lipschitz continuous but not differentiable in the usual sense at some points. Other functions are even more nonsmooth in the sense that they also are discontinuous—for example, in simulating a system that undergoes a discrete change of state. We discuss such functions and the application of direct search methods to them in sections 6.2 and 6.3.

But the problems that interest us here, and the problems to which direct search methods are most often applied, are those just discussed. A serious complication that arises in optimization using complex situations is that, since the associated functions are not expressed in algebraic or analytical form, the computed results may resemble the plot in Figure 1.2, which was modified from [32]. The example involved is a shape optimization problem for viscous channel flow, introduced in [50] to study spurious local minima. The objective is to find a shape parameter to minimize the difference

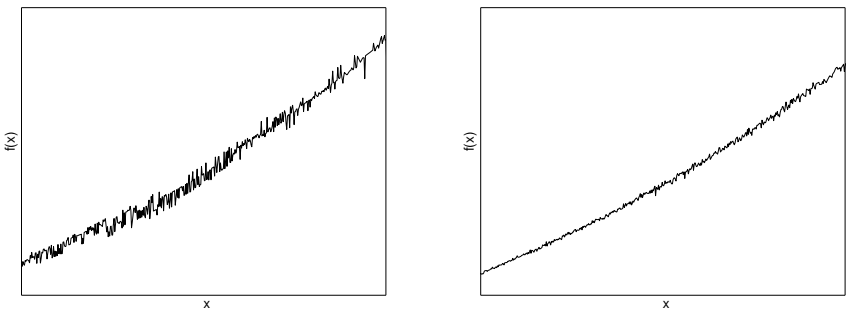


Fig. 1.2 *An objective afflicted with numerical noise deriving from an adaptive finite element scheme. Two adaptations are depicted. Modified from* [32].

between straight channel flow and obstructed channel flow. The underlying infinite-dimensional problem is smooth. The low-amplitude, high-frequency oscillations seen in Figure 1.2 are due to the adaptive finite element scheme used to solve the stationary Navier–Stokes equations. The oscillations diminish with successive adaptations, but the computed objective never becomes smooth, even near the minimizer.

It is widely appreciated in the simulation-based optimization community that the results of complex calculations like those shown in Figure 1.2 may fail to have the level of precision necessary for a reliable finite-difference approximation to the gradient, ruling out an off-the-shelf finite-difference quasi-Newton code. In this example, derivative estimates with a small finite-difference interval are wildly inaccurate.

In general, features such as adaptive algorithms, `if-then-else` logic, stopping tests in iterative schemes inside the simulation, and the inevitable effects of floating point arithmetic are among the culprits that cause smooth problems to appear to be nonsmooth. In fact, many of the approaches employed to improve simulation codes solely for the purposes of simulation, such as mesh adaptivity, contribute to the noise when these simulations are used in connection with optimization. In such a setting the noise is not stochastic in nature. When the simulation is rerun with the same input values, the same output values are returned. Instead, the noise is numerical in nature; it is an inherent feature of the computational problem.

Other forms of noise occur when the function involves limited-precision in the observed data, stochastic variation, or unpredictable, unrepeatable fluctuations from (for example) physical experiments. (Direct search methods trace their origins to such problems; for instance, see the discussion of chemical plant management in [35].) While there has been some analysis of direct search methods for problems with stochastic error [6, 173, 260], we focus on problems with noise that is numerical in nature.

Thus, while in principle it may be possible to obtain derivatives analytically, using either automatic differentiation or finite-difference approximations, certain inherent qualities of simulation-based optimization problems can still make such attempts impractical.

1.2.2. Nonnumerical Functions. Another class of problems to which direct search methods can be applied (and derivative-based methods cannot) involves func-

tions that are not numerical in nature. Instead, the user can only compare objective values to decide which are better and which are worse than the best previously seen. A unique merit of direct search methods—in fact, a motivation for some of the early methods [35]—is their usefulness in these circumstances.

An example arises in the design of digital hearing aids described by Bas Franck of the Department of Experimental and Clinical Audiology, Academic Medical Centre, Amsterdam [112]. There are three variables whose settings are controlled: noise reduction, phonemic compression, and spectral enhancement. In this three-dimensional design space, the goal is to find an optimal setting based on listening comfort and speech intelligibility. These criteria are evaluated and compared for different settings by human beings listening to spoken sentences. Direct search methods are useful for such problems because they need only to identify better and worse points.

1.3. Issues When Using Direct Search Methods. As just discussed, direct search methods have a special niche in modern optimization. In this section we note two issues often cited as unavoidable concerns for any direct search method.

1.3.1. Slow Asymptotic Convergence. No one would disagree with Davidon's comment (quoted in section 1.1) that compass search is slow. As we shall see later (section 3.11), direct search methods are, in a precise sense, asymptotically slower than the steepest descent method. Nonetheless, asymptotic convergence rates do not tell the whole story, especially in the context of the problems to which direct search methods are most applicable.

A common situation in practice is that one wants "improvement" rather than full-blown optimality. The user's goal may be only one or two correct digits, either because this is good enough for the application, or else because the values that can be obtained for the function to be optimized are sufficiently inaccurate that seeking higher accuracy from a solution would be pointless. The function illustrated in Figure 1.2 shows just such an example. In this and other similar situations, the better asymptotic convergence rate of a quasi-Newton method provides no assurance that it will reach a suitable low-accuracy solution more quickly (i.e., with fewer function evaluations) than a direct search method. In fact, the point of the example in Figure 1.2 is that it is quite possible that a quasi-Newton method with finite-difference gradients may converge quickly—to one of the spurious local minimizers, thus returning a meaningless "solution."

A further point to be considered in assessing "slowness" is whether this means only the elapsed time for a computer run, or the total time needed by the user to formulate the problem, write code, and obtain an answer. If human time counts as much as or more than computer time, if the user cannot apply a modeling language or automatic differentiation, and if a modest improvement in the function is acceptable, a direct search method may well produce the desired result more quickly than any alternative. The point here is to note the need, in practical settings, to account for the wall-clock or calendar time that elapses from the time a project is begun to the time usable solutions are obtained.

1.3.2. Limitations on Problem Size. Even in the early days of direct search methods, it was asserted that they were best suited for problems with a small number of variables. We now have some clues as to why performance could deteriorate as the number of variables increases (see section 3.4.1). Even so, direct search methods have been used successfully on problems with a few hundred variables (e.g., [180]). Regardless of dimensionality, they may be the only option in some cases. An obvious

topic for further investigation is the development of approaches to ameliorate this inefficiency.

1.4. What's in a Name? Thus far we have avoided defining what, precisely, is meant by *direct search*. For reasons that will soon become apparent, it is not easy to give an exact definition of this term. We have long been partial to the description given in the 1961 paper by Hooke and Jeeves [139] which, to the best of our knowledge, is where the term "direct search" first appeared:

> We use the phrase "direct search" to describe sequential examination of trial solutions involving comparison of each trial solution with the "best" obtained up to that time together with a strategy for determining (as a function of earlier results) what the next trial solution will be. The phrase implies our preference, based on experience, for straightforward search strategies which employ no techniques of classical analysis except where there is a demonstrable advantage in doing so.

In keeping with the spirit of this passage, we agree with a necessary condition proposed by M. Wright [272], which says, "A direct search method does not 'in its heart' develop an approximate gradient." This excludes finite-difference quasi-Newton schemes, for instance.

The reader might rightly ask for a definition of a mathematical nature. Let us return to Hooke and Jeeves, who give a more formal definition of direct search. The salient features are the following:

1. There is assumed to be an order relation $\prec$ between any two points x and y. For instance, in unconstrained minimization, points x and y may be compared as follows: $x \prec y$ if $f(x) < f(y)$. That is, x is "better" than y because it yields a lower objective function value.
2. At any iteration, only a finite number of possible new iterates exists and the possibilities can be enumerated in advance.

The first requirement insists upon discernible improvement; it is not sufficient merely to match the current value of $f(y)$ (i.e., if $f(x) = f(y)$, then x is not "better" than y and so $x \not\prec y$). Revisiting the illustration of compass search in Figure 1.1 helps make the essence of the second requirement clearer. At each iteration of compass search (in two dimensions), there are exactly four possible steps to consider: one each to the East, West, North, and South. This is true regardless of the function f. Hooke and Jeeves contrast this with steepest descent or Newton's method, where a priori there is a continuum of possible steps. Given an enumeration of possible new iterates, direct search looks among them for a "better" point. So, if our current best point is the iterate x_k, then we want to find a new iterate x_{k+1} for which $x_{k+1} \prec x_k$.

Hooke and Jeeves do not assume that the order relation $\prec$ necessarily involves the comparison of numerical function values. This leads to another take on the meaning of direct search. In Gill, Murray, and Wright [117], the authors remark that, "Methods based on function comparison are often called *direct search methods*." Likewise, Trosset [259] gives the provisional definition, "A direct search method for numerical optimization is any algorithm that depends on the objective function only through the ranks of a countable set of function values." A salient feature of many of the direct search algorithms is the fact that they can be used in the absence of numerical function values, as we saw in the hearing aid example in section 1.2.2. The ability to compare and rank alternatives suffices.

However, the preceding attempts to define direct search have their limitations. For instance, Trosset found his definition did not accommodate stochastic variants of direct search, which led him to a more elaborate definition in [260]. It also is not

clear how one can extend such a simple characterization of direct search algorithms to problems with nonlinear constraints if one wishes to use a merit function such as the augmented Lagrangian (e.g., see section 8.3.2).

To further confuse matters, recently the term "derivative-free optimization" has been used with some frequency in the nonlinear programming literature to mean a number of methods that rely on derivative-free local models of the objective and constraints [71, 74, 177]. These models are derivative-free in the sense that they are not Taylor's series models constructed using analytical or accurate finite-difference estimates of derivatives. Instead, the models are constructed via least-squares fits [121] or interpolation techniques [72, 74, 220]. As if this were not confusing enough, the term "derivative-free" appears in the titles of other papers [115, 173, 174, 175] in reference to algorithms that do *not* explicitly construct a local model of the functions involved.

A precise definition of "direct search" eludes us. Rather than puzzling further over what does, or does not, constitute a direct search method, we move on to a discussion of the objectives of this review.

1.5. Objectives of This Review. We have argued—convincingly, we hope—that direct search methods provide a respectable choice for solving difficult, important optimization problems. Now let us recall why direct search methods were exiled by the mathematical optimization community, as summarized by the quotation from Swann in section 1: the search techniques were based on heuristics, the methods were slow to converge, and there was no mathematical analysis to accompany them. In the course of this review we address each of these points.

Our main goal is a unified presentation of a large number of direct search methods and an explanation of how and why they have the same sorts of theoretical guarantees of convergence as methods that rely explicitly on derivatives. The analysis identifies those heuristics that rest on solid mathematical principles. The analysis also points toward the features of direct search methods that may help explain the slow asymptotic convergence behavior as well as the limitations on the size of the problems these methods can tackle. Furthermore, the analysis hints at why other direct search methods sometimes fail in practice.

Once we have established the basic analytic features of these methods, and discussed some of the interesting algorithmic options, we consider the effects of smoothness (and its lack) on convergence properties.

At the end, we present extensions to problems with bound, linear, and nonlinear constraints. Effective techniques for handling linear and nonlinear constraints remain one of the most active research areas for direct search methods since most simulation-based optimization problems are defined with constraints. Analytically sound methods and good software implementations are now of particular interest.

We have tried to present the material in a way that does not presume a familiarity with optimization on the part of the user. In particular, we try to sketch those elements of nonlinear programming that are needed as background. We also admit to a fondness for pictures; many of the heuristics and intuitions of the original papers on direct search methods derive from pictures of how an optimization algorithm might operate.

1.6. Topics Outside the Scope of This Review. This review will make only passing references to software for direct search methods, though this obviously is an important topic. Except in passing, we do not discuss the parallel implementation of direct search methods. Many direct search methods are amenable to parallelization

[93, 115, 140, 256], but it takes considerable effort to construct implementations that effectively handle load-balancing [140], which can be a serious consideration when either solving simulation-based optimization problems or introducing some of the strategies we discuss in section 8 for handling constraints. Analysis of at least one parallel method [161] introduces some interesting perturbations on the ideas developed here, but we refer the reader to the discussion in [162].

We mention here only briefly work that creates hybrid algorithms containing features of direct search methods. In particular, suggestions have been made to combine direct search methods with a trust region approach [141] and to use direct search methods to obtain convergence in evolutionary algorithms [131, 132, 133]. Various acceleration techniques based on modeling also have appeared [30, 31, 91, 100, 261].

All of these ideas touch in interesting ways on the ideas discussed here, but limitations on length prevent us from giving them a full treatment.

2. Smooth Unconstrained Minimization. For future reference, we give a formal statement of the *unconstrained minimization problem*. Constrained optimization is discussed in sections 7 and 8. The unconstrained problem is

$$\text{minimize} \quad f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$. The function f is the *objective* function. The *variables* x are those directly manipulated by an optimization algorithm. The *dimension* of the problem is n, the number of variables.

2.1. Global Convergence and Local Convergence. In optimization, *order* generally refers to the order of associated derivatives. Hence methods that use only function values (e.g., compass search) are referred to as *zeroth-order methods*, while methods that also use gradients (e.g., steepest descent) are referred to as *first-order methods*.

Similarly, a *first-order optimality condition* is one that involves the gradient. In the case of the unconstrained minimization of a differentiable function f, a necessary (and first-order) condition for a point x_* to be a local minimizer is that $\nabla f(x_*) = 0$, i.e., x_* is a *stationary point* of the function f. *First-order convergence* of an optimization method means that one (or some, or all) of the limit points of the iterates is a stationary point of f.

In the nonlinear programming literature, *global convergence* is used to mean first-order convergence from an arbitrary starting point. In contrast, *local convergence* is used to mean convergence when the initial point is close enough to a minimizer. In addition, local convergence is normally associated with a *rate* of convergence.

Our concern will be mostly with first-order global convergence of direct search methods, though we will obtain a rate of convergence result as a by-product of the analysis. We emphasize that first-order convergence does not imply convergence to a point satisfying second-order conditions, such as that the Hessian matrix is positive definite or semidefinite. Nor does global convergence mean convergence to a global minimizer (i.e., a point x_* such that $f(x_*) \leq f(x)$ for *all* $x \in \mathbb{R}^n$).

In general, it is impossible to prove second-order convergence results (i.e., convergence to a minimizer) for first-order methods like quasi-Newton. Though in practice these algorithms reliably find local minimizers, the global convergence analysis of such methods only establishes first-order convergence. A major point of this review is that similar first-order convergence results hold for a particular class of zeroth-order methods.

2.2. Line Search Methods. Since there will be useful comparisons with what is to follow, a brief sketch of line search methods is included for readers not familiar with them. For further discussion, see texts such as [192, 197] or the survey article [196].

If f is differentiable at x, then a vector $d \in \mathbb{R}^n$ is a *descent direction* for f at x if it is within 90° of $-\nabla f(x)$, i.e.,

$$-\nabla f(x)^T d > 0. \tag{2.1}$$

The definition of the derivative says that

$$f(x + \alpha d) = f(x) + \alpha \nabla f(x)^T d + o(\alpha).$$

If d is a descent direction, and $\alpha > 0$ is sufficiently small, then $x_+ = x + \alpha d$ reduces the value of the objective f. This observation forms the basis of *line search* methods: at the iterate x_k, choose a descent direction d_k and search along this direction for a point $x_{k+1} = x_k + \alpha_k d_k$, with $\alpha_k > 0$, that has a smaller objective value.

In the method of steepest descent, the search direction is $d_k = -\nabla f(x_k)$. One can also incorporate curvature information. Given a positive definite matrix B_k, let

$$d_k = -B_k^{-1} \nabla f(x_k). \tag{2.2}$$

Newton's method corresponds to the choice $B_k = \nabla^2 f(x_k)$, while approximations to the Hessian lead to a variety of secant update quasi-Newton methods.

It would be satisfying to say that the iterates in the algorithm sketched above are guaranteed, at the very least, to converge to a stationary point of f. However, this is not the case. A (perhaps counterintuitive) fact is that *simple decrease* in the objective, i.e.,

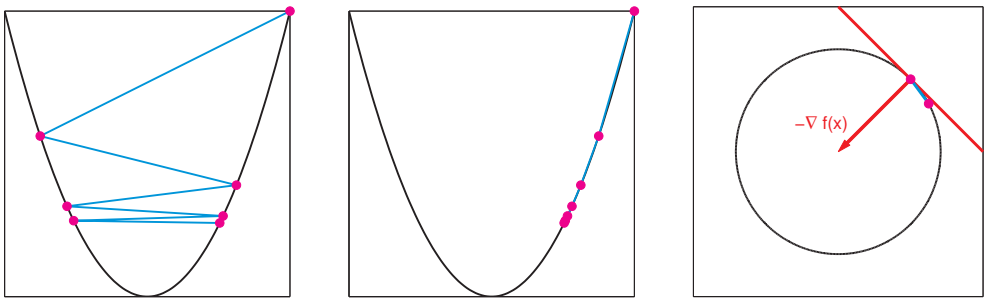
$$f(x_{k+1}) < f(x_k), \tag{2.3}$$

is not enough to ensure convergence to a stationary point of f.

Two reasons for this are illustrated by the following example from [92]. Figure 2.1(a)–(b) shows two sequences of iterates for which $f(x_{k+1}) < f(x_k)$, where $f(x) = x^2$. However, neither sequence converges to 0, the only stationary point of f. In (a), the sequence is $x_k = (-1)^k(1 + 2^{-k})$, with limit points at ± 1. In (b), the sequence is $x_k = 1 + 2^{-k}$, which converges to 1. In (a), the problem is that the steps are "too long" relative to the amount of decrease seen from one iterate to the next. In (b), the steps are "too short" relative to the linear rate of decrease in the function.

Figure 2.1(c) illustrates yet another potential problem. The level curve of the function at x_k is shown in black. The direction of steepest descent from x_k is indicated by the red arrow; the tangent plane at x_k is indicated by the red line. The direction d_k, indicated by the blue line segment, satisfies the descent condition (2.1), but just barely, as it nearly forms a right angle with the direction of steepest descent. As a consequence, only short steps along d_k will yield improvement; in the picture, only the region indicated by the blue line segment. If the angle between d_k and $-\nabla f(x_k)$ approaches 90°, the iterates may converge prematurely to a point that is not a stationary point.

These examples may seem terribly contrived. But it turns out that the only two pathologies to guard against in order to be assured of convergence to a stationary point are poor choices of step length and poor choices of descent directions.



(a) Steps are too long. (b) Steps are too short. (c) Bad search direction.

Fig. 2.1 *The ways that a line search algorithm can fail.*

2.3. Avoiding Poor Choices of Step Lengths. In Figure 2.1(a)–(b), each step is along a direction of descent, but the steps yield an ever decreasing fraction of the total improvement in $f(x)$ promised by the derivative. Traditionally, this has been corrected by imposing acceptance criteria that tie the step-length control parameter α_k to the expected decrease as estimated by the initial rate of decrease $-\nabla f(x_k)^T d_k$. Specifically, the condition

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T d_k \tag{2.4}$$

prevents steps that are too long via a *sufficient decrease* criterion, while the condition

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq c_2 \nabla f(x_k)^T d_k \tag{2.5}$$

prevents steps that are too short via a curvature criterion, with $0 < c_1 < c_2 < 1$. The first criterion (2.4) is sometimes called the Armijo condition [9, 192]. A related criterion is the Goldstein condition [125, 197]. The second criterion (2.5) is sometimes called the Wolfe condition [192, 269]. Collectively, (2.4) and (2.5) may be referred to as the Armijo–Goldstein–Wolfe conditions.

To see how (2.4) prevents steps that are too long, consider the example in Figure 2.2. Rewriting (2.4) yields

$$f(x_k + \alpha_k d_k) - f(x_k) \leq -c_1 \alpha_k |\nabla f(x_k)^T d_k|. \tag{2.6}$$

Each side of the inequality can be parameterized, in terms of the step length $\alpha \geq 0$, as

$$g(\alpha) = f(x_k + \alpha d_k) - f(x_k) \quad \text{and} \quad h(\alpha) = -c_1 \alpha |\nabla f(x_k)^T d_k|.$$

In Figure 2.2, any choice of $\alpha_k < \bar{\alpha}$ will satisfy (2.6). The requirement on α_k is that it produces decrease in the value of f at x_k that is at least some fraction of the decrease predicted by the linear approximation. When c_1 is chosen to be near zero (i.e., much less than 1), the sufficient decrease condition is easier to satisfy since only a small decrease in the function value is required.

Now observe that $g(0) = h(0) = 0$. If f is continuously differentiable in a neighborhood of x_k and d_k is a descent direction, then $g'(0) = \nabla f(x_k)^T d_k < 0$. Furthermore, $h'(0) = c_1 \nabla f(x_k)^T d_k < 0$. Since $c_1 < 1$, there is guaranteed to be a range of

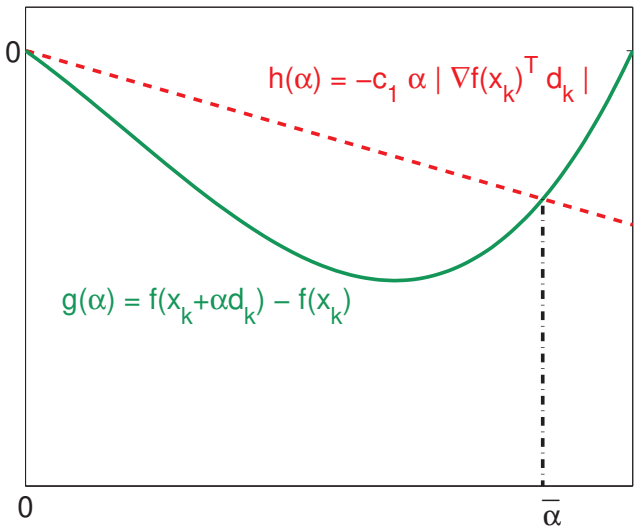


Fig. 2.2 *Illustration that suitably small step lengths will satisfy the sufficient decrease condition.*

choices for α for which $g(\alpha) < h(\alpha)$ holds. The important observation here is that (2.4) ensures that when f is continuously differentiable and d_k is a descent direction, then there exists a nonempty interval $(0, \bar{\alpha})$ of acceptable α_k values.

Condition (2.5), which guards against steps that are "too short," is frequently not enforced directly in practice. Instead, this condition is satisfied automatically using a technique known as *backtracking*. In a backtracking line search, the search begins with a relatively long step and only tries shorter steps, as needed, to find an acceptable step (i.e., one satisfying (2.4)).

2.4. Avoiding Poor Choices of Descent Directions. The direction d_k is a descent direction at x_k if it satisfies (2.1). However, as seen in Figure 2.1(c), the algorithm can stall if d_k is almost orthogonal to the direction of steepest descent. This can be prevented by enforcing a uniform lower bound on the angle between $-\nabla f(x_k)$ and d_k:

$$\frac{-\nabla f(x_k)^T d_k}{\| \nabla f(x_k) \| \, \| d_k \|} \geq c > 0, \tag{2.7}$$

where c is independent of k. Methods for which condition (2.7) holds often are referred to as *gradient-related* [202]. Alternatively, (2.7) may be referred to as the *angle condition* [192] since it also can be written as $\cos\theta \geq c > 0$, where θ is the angle between the search direction d_k and the direction of steepest descent $-\nabla f(x_k)$. Note that if d_k and $-\nabla f(x_k)$ are orthogonal, then $\cos\theta = 0$. Condition (2.7) automatically holds in the method of steepest descent.

2.5. A Global Convergence Result for Line Search Methods. The following is a standard first-order global convergence result for line search methods [92, 192, 197].

THEOREM 2.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable on $\mathbb{R}^n$ and be bounded below. Furthermore, suppose that ∇f is Lipschitz continuous with constant*

M; that is,

$$\| \nabla f(y) - \nabla f(x) \| \leq M \| y - x \| \quad \textit{for all } x, y \in \mathbb{R}^n.$$

If the sequence $\{x_k\}$ satisfies the step-length conditions (2.4)–(2.5) *and the search direction condition* (2.7)*, then*

$$\lim_{k \to \infty} \| \nabla f(x_k) \| = 0.$$

Note that this result does *not* say that the sequence necessarily converges to a stationary point. Rather, it says that any limit point of the sequence of iterates is a stationary point. Further assumptions are required in order to obtain a guarantee of convergence of the entire sequence. Typically, one assumes that one of the limit points x_* is a point satisfying the second-order sufficiency condition $\nabla^2 f(x_*) > 0$. Then f is convex near x_*, i.e., its graph is bowl-shaped. One then shows that once x_k enters some neighborhood of x_*, further iterates will not leave this neighborhood, and hence the entire sequence will converge to x_*.

3. Generating Set Search Methods for Unconstrained Minimization. As intimated in the introduction, not all direct search methods reliably find solutions. Some algorithms, such as the simplex algorithm of Nelder and Mead, sometimes find minimizers very efficiently, but they can also fail unpredictably. On the other hand, other algorithms, such as compass search or the pattern search method of Hooke and Jeeves, have been observed to be very reliable, if sometimes slow, in practice.

For this reason, much of this review focuses on identifying a class of direct search methods which can be shown to have a mathematical foundation as sound as that for, say, derivative-based line search methods. Of particular interest is the articulation of the features that make some direct search methods reliable, in the sense of having attendant convergence results similar to those for line search methods. Such an examination sheds light on why direct search methods behave as they do, in terms of their robustness, dependence of performance on dimension, local rate of convergence, and the like. It also shows the rich algorithmic possibilities for new classes of direct search methods.

Because there is no name for the particular class of direct search methods that is our focus, we introduce the name *generating set search* (GSS). We hope that the benefits of a clearly defined class outweigh the annoyance of introducing yet another name to the nomenclature.

As we shall see, in unconstrained optimization it is crucial for the set of search directions to be a positive spanning set for $\mathbb{R}^n$, meaning that every vector in $\mathbb{R}^n$ can be written as a linear combination of these directions with nonnegative weights. This is made formal in Definition 3.1 of section 3.4. This suggests the name "positive spanning set methods." While this description is valid in the case of unconstrained optimization, it does not convey the properties needed by the set of search directions when optimizing subject to bounds and general linear (or linearized) constraints. In these cases, as discussed in section 8, the search directions need to generate a cone defined by the "almost active" constraint normals, and thus we propose the name "generating set search methods." This name applies as well for unconstrained problems, where the search directions must be the generators for $\mathbb{R}^n$, interpreted as a cone. Generalization to the constrained case is covered in section 8.

GSS includes the generalized pattern search methods previously analyzed by Torczon [257], which cover as special cases the pattern search algorithm of Hooke

and Jeeves [139], multidirectional search [256], variants of the EVOP technique introduced by Box [35], our old friend compass search, as well as the extensions introduced by Lewis and Torczon [166] to handle positive bases. Additionally, GSS includes a subset of the algorithms encapsulated in the grid-based framework of Coope and Price [76, 77]. All of the preceding methods enforce only the simple decrease condition (2.3). GSS also includes methods based on sufficient decrease conditions similar to (2.4), but which do not use derivative information. Algorithms of this sort include those of Yu [278], Lucidi and Sciandrone [174], and García-Palomares and Rodríguez [115].

The goal here is to design GSS so that it is generic enough to capture many of the algorithmic variations in the literature while at the same time being simple enough to discuss with a minimum of notation and without handling a variety of special cases. Thus, for now, we deliberately suppress some of the algorithmic possibilities allowed in the work just mentioned. In section 5, we will discuss a few of the interesting variations that are not neatly covered by our abstraction.

The discussion of derivative-based line search methods in section 2.2 introduced the principles that will be used in proving convergence for GSS methods. First, the algorithm must have a search direction that is a descent direction, as defined in (2.1). Second, GSS methods must avoid poor search directions; i.e., there must be a search direction that is not "too orthogonal" to the direction of steepest descent, as in (2.7). GSS methods use multiple search directions—specifically a *generating* set—to ensure that (2.1) and (2.7) are satisfied by at least one direction in the set, even without explicit knowledge of the gradient. Finally, GSS methods must avoid poor choices of step lengths. Since all the GSS methods employ a backtracking strategy, there is no need to enforce (2.5). The novelty in the analysis of GSS methods lies in the way in which they *replace* (2.4)—since they do not have an explicit representation of $\nabla f(x_k)$ to use—and yet still manage to avoid taking steps that are too long. Once it is clear that GSS methods satisfy these basic conditions, the necessary ingredients will be available to prove results analogous to Theorem 2.1.

3.1. An Example: Compass Search Revisited. Returning to the discussion of compass search from section 1.1, we formalize the terminology and notation. The method is stated in Algorithm 3.1. Let k serve as the index for each *iteration*. Let $x_k \in \mathbb{R}^n$ denote the kth *iterate*, with x_0 denoting the *initial guess*. Let $\mathcal{D}_\oplus$ denote the set of $2n$ *coordinate directions*, defined as the positive and negative unit coordinate vectors

$$\mathcal{D}_\oplus = \{e_1, e_2, \ldots, e_n, -e_1, -e_2, \ldots, -e_n\}.$$

Let Δ_k denote the *step-length control parameter* that controls the lengths of the steps taken. Initialization of the algorithm includes choosing a starting value Δ_0. In Figure 1.1 we use $\Delta_0 = 0.3$.

For applications where the objective value is expensive to compute, the majority of the work occurs in Step 2, the evaluation of the objective at *trial points* of the form $x_k + \Delta_k d$ for each $d \in \mathcal{D}_\oplus$.

There is tremendous latitude in exactly how one does the evaluations to determine if there exists a $d_k \in \mathcal{D}_\oplus$ satisfying the simple decrease condition (2.3). For example, one may evaluate f at each of the $2n$ trial points in sequence and choose the trial point that yields the greatest decrease in f. Alternatively, if a parallel computer is available, one may perform the $2n$ objective evaluations simultaneously, and then choose the trial point that yields the most decrease. Another option is to evaluate each trial point

Initialization.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be given.

Let $x_0 \in \mathbb{R}^n$ be the initial guess.

Let $\Delta_{\text{tol}} > 0$ be the tolerance used to test for convergence.

Let $\Delta_0 > \Delta_{\text{tol}}$ be the initial value of the step-length control parameter.

Algorithm. For each iteration $k = 1, 2, \ldots$

Step 1. Let $\mathcal{D}_\oplus$ be the set of coordinate directions $\{\pm e_i \,|\, i = 1, \ldots, n\}$, where e_i is the ith unit coordinate vector in $\mathbb{R}^n$.

Step 2. If there exists $d_k \in \mathcal{D}_\oplus$ such that $f(x_k + \Delta_k d_k) < f(x_k)$, then do the following:

- Set $x_{k+1} = x_k + \Delta_k d_k$ (change the iterate).
- Set $\Delta_{k+1} = \Delta_k$ (no change to the step-length control parameter).

Step 3. Otherwise, $f(x_k + \Delta_k d) \geq f(x_k)$ for all $d \in \mathcal{D}_\oplus$, so do the following:

- Set $x_{k+1} = x_k$ (no change to the iterate).
- Set $\Delta_{k+1} = \frac{1}{2}\Delta_k$ (contract the step-length control parameter).
- If $\Delta_{k+1} < \Delta_{\text{tol}}$, then **terminate**.

Algorithm 3.1 *Compass search: an example of a generating set search method.*

in sequence but stop at the first one, if any, that satisfies the decrease condition (2.3). Whenever a direction satisfying (2.3) is found, we say that the iterate is *successful*. The direction that produced success is denoted by d_k, and the subsequence of successful iterations is denoted by $\mathcal{S}$.

Regardless of the procedure chosen for evaluating the trial points, the value of Δ_k is not reduced unless every trial point has been evaluated and found unacceptable. In this case, none of the $2n$ trial points $x_k + \Delta_k d$, $d \in \mathcal{D}_\oplus$, produces decrease. Such an iteration is called *unsuccessful* and the algorithm proceeds to Step 3, where the step-length control parameter is halved. The subsequence of unsuccessful iterations is denoted by $\mathcal{U}$.

Furthermore after any unsuccessful iteration (i.e., where Δ_k is reduced), Δ_k is compared to the preset stopping tolerance Δ_{tol} to test for convergence. Once the step-length control parameter falls below Δ_{tol}, the search terminates with $x_* = x_{k+1}$. Note that Δ_k is only shortened after unsuccessful steps; thus the search will not stop after a successful step. There are good reasons for enforcing this particular convention, as the discussion of convergence in section 3.6 will make clear. Relaxations of this convention are possible, as discussed in section 5.1.

3.2. Convergence of Compass Search. Consider the coordinate directions in $\mathcal{D}_\oplus$. As depicted for two dimensions in Figure 1.1, it is easy to see that at each

iteration, at least one of the four coordinate directions must be a descent direction. In fact, one of the four coordinate directions must be within 45° of the direction of steepest descent, *no matter what the direction of steepest descent might be.* Something similar is true for any dimension n: given any $x \in \mathbb{R}^n$ for which $\nabla f(x) \neq 0$, at least one of the coordinate directions must be a descent direction. Looking in *multiple* directions implicitly ensures a descent direction, so long as the current iterate is not a stationary point of f.

Now suppose at iteration k, a step $\Delta_k > 0$ along each of the coordinate directions was taken but none of those steps produced decrease; i.e.,

$$f(x_k) \leq f(x_k + \Delta_k\, d) \text{ for all } d \in \mathcal{D}_{\oplus}. \tag{3.1}$$

This is an unsuccessful iteration. What occurs at unsuccessful iterations is key to proving convergence results for compass search and, more generally, GSS methods.

In the case of the coordinate directions in compass search, the cosine of the largest angle between an arbitrary vector v and the closest coordinate direction is bounded below by $\frac{1}{\sqrt{n}}$ [257]. This means that no matter the value of $\nabla f(x)$ there is at least one $d \in \mathcal{D}_{\oplus}$ for which

$$\frac{1}{\sqrt{n}} \,\|\, \nabla f(x_k) \,\|\, \|\, d \,\| \leq -\nabla f(x_k)^T d. \tag{3.2}$$

This is nothing other than (2.7) from our discussion of line search methods, with $c = 1/\sqrt{n}$.

If iteration k is unsuccessful as in (3.1), then the mean value theorem says

$$0 \leq f(x_k + \Delta_k\, d) - f(x_k) = \nabla f(x_k + \alpha_k \Delta_k d)^T\, \Delta_k d$$

for some $\alpha_k \in [0, 1]$. Subtracting $\Delta_k \nabla f(x_k)^T d$ from both sides leads to

$$-\Delta_k \nabla f(x_k)^T d \leq \Delta_k (\nabla f(x_k + \alpha_k \Delta_k d) - \nabla f(x_k))^T d.$$

Applying (3.2) yields

$$\frac{1}{\sqrt{n}} \,\|\, \nabla f(x_k) \,\|\, \|\, d \,\| \leq (\nabla f(x_k + \alpha_k \Delta_k d) - \nabla f(x_k))^T d.$$

Now suppose f is continuously differentiable and, to keep things simple, that ∇f is Lipschitz with constant M (uniform continuity would do). The role of differentiability, and the degree to which it is (and is not) restrictive with regard to nonsmooth or noisy objectives in theory and in practice, is discussed at length in section 6.

Under these assumptions,

$$\frac{1}{\sqrt{n}} \,\|\, \nabla f(x_k) \,\|\, \|\, d \,\| \leq M \,\|\, \alpha_k \Delta_k d \,\|\, \|\, d \,\| \leq M\, \Delta_k \| \, d \,\|^2,$$

or, since $\|\, d \,\| = 1$,

$$\|\, \nabla f(x_k) \,\| \leq \sqrt{n}\, M\, \Delta_k. \tag{3.3}$$

Thus, even though there is no explicit knowledge of the gradient, *at unsuccessful iterations* there is an implicit bound on the norm of the gradient in terms of the step-length control parameter Δ_k.

If, in addition, it could be shown that

$$\lim_{\substack{k \to +\infty \\ k \in \mathcal{U}}} \Delta_k = 0,$$

then it would be possible to conclude that

$$\lim_{\substack{k \to +\infty \\ k \in \mathcal{U}}} \|\nabla f(x_k)\| \leq \lim_{\substack{k \to +\infty \\ k \in \mathcal{U}}} \sqrt{n}\, M\, \Delta_k = 0.$$

Thus, any convergent sequence of unsuccessful iterates must converge to a point where the gradient vanishes. This is a convergence result analogous to Theorem 2.1 for line search methods, which explicitly use the gradient.

The crux of the convergence analysis for GSS methods will be to show that, in fact,

$$\lim_{\substack{k \to +\infty \\ k \in \mathcal{U}}} \Delta_k = 0.$$

Even though it is at successful iterations that f is reduced, it is the unsuccessful iterations that are key to proving convergence.

3.3. Generating Set Search. The ideas underlying compass search are expanded in the generic GSS method given in Algorithm 3.2. Our goal is to state the algorithm broadly enough to encompass many of the methods in the literature so that we can identify the common elements. Significant changes from the compass search algorithm given in Algorithm 3.1 are highlighted in red. There are three major changes to note.

The first major change regards the set of search directions used in Step 1. Previously the coordinate directions were the only search directions. This is replaced by a set of search directions $\mathcal{D}_k$ which contains a *generating set* $\mathcal{G}_k$ for $\mathbb{R}^n$. Generating sets are defined formally in Definition 3.1 in section 3.4; the set of coordinate directions is one example of a generating set. It is also possible to include additional directions for the search in $\mathcal{H}_k$. For compass search, $\mathcal{H}_k = \emptyset$ and $\mathcal{G}_k = \mathcal{D}_{\oplus}$. The set of directions $\mathcal{D}_k$ may change at each iteration (hence the introduction of the subscript k on the set $\mathcal{D}$). However, the *cosine measure* for the generating set $\mathcal{G}_k$, which is defined formally in (3.10) in section 3.4.1 and denoted by $\kappa(\mathcal{G}_k)$, must be bounded below. The cosine measure plays the role of (2.7) in line search methods.

While the generating set $\mathcal{G}_k$ and the cosine measure $\kappa(\mathcal{G}_k)$ are key to the analysis, the set $\mathcal{H}_k$ is tangential to the analysis. Thus it is acceptable for $\mathcal{H}_k$ to be empty, but it opens up the possibility for interesting heuristic strategies designed to accelerate the progress of the search. We discuss some of these possibilities in section 4.

The second major change concerns the update to the step-length control parameter Δ_k in Steps 2 and 3. For coordinate search, the traditional choices are $\phi_k = 1$ for all $k \in \mathcal{S}$ and $\theta_k = \frac{1}{2}$ for all $k \in \mathcal{U}$. However, at a successful iteration it is possible to *expand* the length of the step by increasing Δ_k (i.e., by allowing $\phi_k > 1$). At unsuccessful iterations it is necessary to *contract* the length of the step by reducing Δ_k (i.e., by requiring $\theta_k \in (0, 1)$), but now there is more flexibility in the choice of θ_k. Note, however, that θ_k is required to be bounded below by 0 and above by some value $\theta_{\max} < 1$, which the user is free to choose during the initialization of the algorithm. Finally, the *expansion factor* ϕ_k and the *contraction factor* θ_k may be changed at each iteration.

The third major change is a modification of the criterion used to accept an iterate (i.e., the way in which "success" is defined). In the definition of compass search given

Initialization.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be given.

Let $x_0 \in \mathbb{R}^n$ be the initial guess.

Let $\Delta_{\text{tol}} > 0$ be the step-length convergence tolerance.

Let $\Delta_0 > \Delta_{\text{tol}}$ be the initial value of the step-length control parameter.

Let $\theta_{\max} < 1$ be an upper bound on the contraction parameter.

Let $\rho : [0, +\infty) \to \mathbb{R}$ be a continuous function such that $\rho(t)$ is decreasing as $t \to 0$ and $\rho(t)/t \to 0$ as $t \downarrow 0$. The choice $\rho \equiv 0$ is acceptable.

Let $\beta_{\max} \geq \beta_{\min} > 0$ be upper and lower bounds, respectively, on the lengths of the vectors in any generating set.

Let $\kappa_{\min} > 0$ be a lower bound on the cosine measure of any generating set.

Algorithm. For each iteration $k = 1, 2, \ldots$

Step 1. Let $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$. Here $\mathcal{G}_k$ is a generating set for $\mathbb{R}^n$ satisfying $\beta_{\min} \leq \|d\| \leq \beta_{\max}$ for all $d \in \mathcal{G}_k$ and $\kappa(\mathcal{D}_k) \geq \kappa_{\min}$, and $\mathcal{H}_k$ is a finite (possibly empty) set of additional search directions such that $\beta_{\min} \leq \|d\|$ for all $d \in \mathcal{H}_k$.

Step 2. If there exists $d_k \in \mathcal{D}_k$ such that $f(x_k + \Delta_k d_k) < f(x_k) - \rho(\Delta_k)$, then do the following:

- Set $x_{k+1} = x_k + \Delta_k d_k$ (change the iterate).
- Set $\Delta_{k+1} = \phi_k \Delta_k$, where $\phi_k \geq 1$ (optionally expand the step-length control parameter).

Step 3. Otherwise, $f(x_k + \Delta_k d) \geq f(x_k) - \rho(\Delta_k)$ for all $d \in \mathcal{D}_k$, so do the following:

- Set $x_{k+1} = x_k$ (no change to the iterate).
- Set $\Delta_{k+1} = \theta_k \Delta_k$ where $0 < \theta_k < \theta_{\max} < 1$ (contract the step-length control parameter).
- If $\Delta_{k+1} < \Delta_{\text{tol}}$, then **terminate**.

Algorithm 3.2 *A GSS method.*

in section 3.1, any step that produces *simple decrease* in $f(x_k)$ is acceptable. In other words, in Step 2, so long as there exists $d_k \in \mathcal{D}_k$ such that $f(x_k + \Delta_k d_k) < f(x_k)$, the iteration is judged a success. The introduction of a *forcing function* $\rho(\cdot)$ now also allows the possibility of enforcing *sufficient decrease* by establishing a threshold for the amount of decrease that must be realized for the iteration to be deemed a success. This generalization allows for the use of *either* the simple decrease criterion *or* the sufficient decrease criterion. In the first case, simply continue with $\rho \equiv 0$. In the second case, use a forcing function that satisfies the condition $\rho(t)/t \to 0$ as $t \to 0$. (A simple choice is $\rho(t) = at^2$ for some $a > 0$.) In either case, global convergence results can be derived, but the choice of decrease criterion has significant bearing on

the conditions that must be placed on $\mathcal{D}_k$, ϕ_k, and θ_k. This is discussed in more detail in section 3.7

The statement of the GSS algorithm given in Algorithm 3.2 by no means captures every useful variation on GSS methods. But it captures the essence of the classical pattern search methods [257] while encompassing many of the more recent ideas found in the nonlinear optimization literature (e.g., [5, 13, 76, 77, 115, 166, 174]).

3.3.1. Bounds on the Lengths of the Directions. Uniform bounds are imposed on the lengths of the directions in the sets of search directions:

$$\begin{array}{ll} \beta_{\min} \le \|d\| \le \beta_{\max} & \text{for all } d \in \mathcal{G}_k, \text{ and} \\ \beta_{\min} \le \|d\| & \text{for all } d \in \mathcal{H}_k, \end{array} \quad \text{for } k = 1, 2, \ldots. \tag{3.4}$$

The length of the step is controlled by the step-length control parameter Δ_k.

3.3.2. Update Formulas. For later reference, we explicitly state the rules for updating the iterate x_k and the step-length control parameter Δ_k. First,

$$x_{k+1} = \begin{cases} x_k + \Delta_k d_k, & k \in \mathcal{S}, \\ x_k, & k \in \mathcal{U}. \end{cases} \tag{3.5}$$

Note that d_k is only defined for successful iterations; i.e., $k \in \mathcal{S}$. Second,

$$\Delta_{k+1} = \begin{cases} \phi_k \Delta_k, & k \in \mathcal{S}, \\ \theta_k \Delta_k, & k \in \mathcal{U}, \end{cases} \tag{3.6}$$

with $\phi_k \ge 1$ and $0 < \theta_k \le \theta_{\max} < 1$.

3.3.3. Step Acceptance Criteria. Recall that an iteration k is *successful*, i.e., $k \in \mathcal{S}$, if it satisfies the *decrease condition*. That is, there exists a $d_k \in \mathcal{D}_k$ for which

$$f(x_k + \Delta_k d_k) < f(x_k) - \rho(\Delta_k). \tag{3.7}$$

The nonnegative function ρ defined on $[0, +\infty)$ is called the *forcing function* and must satisfy one of the following two requirements. Either

$$\rho(t) \equiv 0 \tag{3.8}$$

or

$$\rho \text{ is continuous}, \quad \rho(t) = o(t) \text{ as } t \downarrow 0, \quad \text{and} \quad \rho(t_1) \le \rho(t_2) \text{ for } t_1 < t_2. \tag{3.9}$$

Choosing ρ to be identically zero as in (3.8) imposes a *simple decrease* condition on the acceptance of the step. Otherwise, choosing ρ as in (3.9) imposes a *sufficient decrease* condition.

Though similar, this usage of "forcing function" should not be confused with other definitions of the same term, such as that of Ortega and Rheinboldt [202].

3.4. Generating Sets for $\mathbb{R}^n$. Each iteration of the GSS method given in Algorithm 3.2 requires a set $\mathcal{G}_k$ that is a generating set for $\mathbb{R}^n$.

DEFINITION 3.1. *Let $\mathcal{G} = \{d^{(1)}, \ldots, d^{(p)}\}$ be a set of $p \ge n+1$ vectors in $\mathbb{R}^n$. Then the set $\mathcal{G}$* generates *(or* positively spans*) $\mathbb{R}^n$ if for any vector $v \in \mathbb{R}^n$, there exist $\lambda^{(1)}, \ldots, \lambda^{(p)} \ge 0$ such that*

$$v = \sum_{i=1}^{p} \lambda^{(i)} d^{(i)}.$$

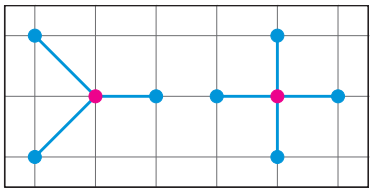


Fig. 3.1 *Examples of generating sets for* $\mathbb{R}^2$ *with* $n+1$ *and* $2n$ *vectors.*

A generating set for $\mathbb{R}^n$ is the same as a *positive spanning set* [86]. Any vector in $\mathbb{R}^n$ can be written as a linear combination of the generators, so a generating set is like a spanning set, except for the additional requirement that all the coefficients be nonnegative.

A generating set must contain a minimum of $n+1$ vectors. In two dimensions, a minimal generating set with $n+1=3$ vectors is

$$\mathcal{G} = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\}.$$

The coordinate directions also form a generating set with $2n$ vectors. In two dimensions, these are the $2n=4$ vectors in the set

$$\mathcal{G} = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right\}.$$

Figure 3.1 illustrates both these examples.

3.4.1. Generating Sets and Descent. Having a set guaranteed to include a descent direction was one of the ingredients in the discussion in section 3.2 of why compass search should work. The generalization to arbitrary generating sets maintains this key feature, as the following result [86] shows.

LEMMA 3.2. *The set* $\mathcal{G}$ generates $\mathbb{R}^n$ *if and only if for any vector* $v \in \mathbb{R}^n$ *such that* $v \neq 0$*, there exists* $d \in \mathcal{G}$ *such that*

$$v^T d > 0.$$

Geometrically, Lemma 3.2 says that $\mathcal{G}$ generates $\mathbb{R}^n$ if and only if the interior of every half-space contains a member of $\mathcal{G}$. The significance of Lemma 3.2 to GSS is that at every iteration k, there must be some $d \in \mathcal{G}_k$ such that

$$-\nabla f(x_k)^T d > 0.$$

This means $\mathcal{G}_k$ is guaranteed to contain a descent direction whenever $\nabla f(x_k) \neq 0$.

Recall also from the discussion of compass search that a bound on the cosine of the maximum angle between any arbitrary vector v and the set of coordinate directions played a role in the analysis. In the case of a general generating set $\mathcal{G}$, we refer to that value as the *cosine measure* of $\mathcal{G}$ and denote it by $\kappa(\mathcal{G})$. Formally, the cosine measure is defined as

$$\kappa(\mathcal{G}) \equiv \min_{v \in \mathbb{R}^n} \max_{d \in \mathcal{G}} \frac{v^T d}{\|v\| \, \|d\|}. \tag{3.10}$$

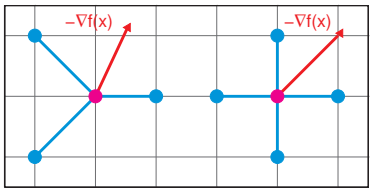


Fig. 3.2 *How far the negative gradient can be from the vectors in these two generating sets for* $\mathbb{R}^2$.

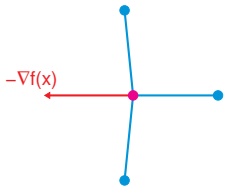


Fig. 3.3 *A generating set with a very small cosine measure.*

Clearly, if $\mathcal{G}$ is a generating set, $\kappa(\mathcal{G}) > 0$. The cosine measure captures how far the steepest descent direction can be, in the worst case, from the vector in $\mathcal{G}$ making the smallest angle with $v = -\nabla f(x)$. This worst-case behavior is illustrated in Figure 3.2 for the two generating sets shown in Figure 3.1.

In terms of descent, this means that for any generating set $\mathcal{G}$, there must exist a $d \in \mathcal{G}$ such that

$$\kappa(\mathcal{G}) \, \|\nabla f(x)\| \, \|d\| \leq -\nabla f(x)^T d. \tag{3.11}$$

Why is the cosine measure important? Suppose that the sequence of generating sets is defined by

$$\mathcal{G}_k = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} -\frac{1}{k} \\ -1 \end{bmatrix}, \begin{bmatrix} -\frac{1}{k} \\ 1 \end{bmatrix} \right\}.$$

Then as $k \to +\infty$, the cosine measure of this set goes to zero; specifically, $\kappa(\mathcal{G}_k) = 1/\sqrt{1+k^2}$. If $-\nabla f(x) = -e_1$, as shown in Figure 3.3, then the quality of the descent directions in $\mathcal{G}_k$ is poor and the lower bound in (3.11) is small relative to $\|\nabla f(x)\|$.

To prevent pathologies such as this, the cosine measure must be bounded below; that is,

$$\kappa(\mathcal{G}_k) \geq \kappa_{\min} \quad \text{for all} \quad k = 1, 2, \ldots. \tag{3.12}$$

This limits how "bad" the search directions can be, which is important to the analysis that follows.

This also helps explain why the performance of GSS methods may deteriorate as n grows. Recall that the cosine measure of the coordinate directions, $\mathcal{D}_\oplus$, decreases as n grows; i.e., $\kappa(\mathcal{D}_\oplus) = 1/\sqrt{n}$.

The concept of a generating set is helpful—both in the analysis *and* in practice—because using a generating set guarantees a direction of descent, even when $\nabla f(x_k)$ is unknown. The drawback, though, is that without an explicit representation of

$\nabla f(x_k)$, there is no way to identify *which* of the vectors in the generating set are guaranteed to be directions of descent, and thus the algorithm may have to search along each of the vectors in the generating set to find decrease in f.

Imposing the condition (3.12) is but one technique to guarantee that the search directions are, in a sense, well conditioned. Another possibility is to impose conditions on the determinants of certain subsets of the generating set, as is done by Yu [278] and Coope and Price [77]. Yet another possibility is to impose a condition on a particular limit involving the search directions, as is done by Lucidi and Sciandrone [174].

3.5. The Ingredients for Proving Global Convergence. The stage is now set for a discussion of the two basic ingredients for proving global convergence for GSS methods.

The first ingredient is showing that for any subsequence $\mathcal{K}$ of unsuccessful iterations (i.e., $\mathcal{K} \subseteq \mathcal{U}$),

$$\lim_{\substack{k\to+\infty \\ k\in\mathcal{K}\subseteq\mathcal{U}}} \Delta_k = 0 \quad \Rightarrow \quad \lim_{\substack{k\to+\infty \\ k\in\mathcal{K}\subseteq\mathcal{U}}} \| \nabla f(x_k) \| = 0.$$

In compass search, this was accomplished by showing that at any unsuccessful iteration, $\|\nabla f(x_k)\|$ was $O(\Delta_k)$; see (3.3). Theorem 3.3 states a general result for GSS methods.

The second ingredient is showing there is indeed a subsequence of step-length control parameters going to zero. We refer to this as *globalization* since this ingredient plays much the same role as a line search in gradient-based methods. Three examples of globalization techniques are analyzed in section 3.7.

In section 3.8, these ingredients are brought together to establish both weak and strong convergence results.

3.6. Relating Δ_k to the Measure of Stationarity. For compass search, (3.3) gave a bound on the norm of the gradient at unsuccessful iterations in terms of the step-length control parameter Δ_k. Theorem 3.3, which follows, gives a general bound for GSS methods.

The following theorem leads not only to first-order convergence results but also to local convergence results and to the validation of a long-standing practical stopping criterion for GSS methods (see section 3.10). For simplicity, we assume $\nabla f(x)$ is Lipschitz, but this is not essential, as noted after the proof.

THEOREM 3.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable, and suppose ∇f is Lipschitz continuous with constant M. Then GSS produces iterates such that for any $k \in \mathcal{U}$, we have*

$$\| \nabla f(x_k) \| \leq \kappa(\mathcal{G}_k)^{-1} \left[M\Delta_k\beta_{\max} + \frac{\rho(\Delta_k)}{\Delta_k\beta_{\min}} \right].$$

Proof. Choose $\hat{d}_k \in \mathcal{G}_k \subseteq \mathcal{D}_k$ satisfying (3.11). Such a $\hat{d}_k$ exists because $\mathcal{G}_k$ generates $\mathbb{R}^n$; so,

$$\kappa(\mathcal{G}_k) \, \| \nabla f(x_k) \| \, \| \hat{d}_k \| \leq -\nabla f(x_k)^T \hat{d}_k.$$

By the mean value theorem, for some $\alpha_k \in [0, 1]$,

$$f(x_k + \Delta_k\hat{d}_k) - f(x_k) = \Delta_k \nabla f(x_k + \alpha_k\Delta_k\hat{d}_k)^T \hat{d}_k.$$

Because k is an unsuccessful iteration,

$$0 \leq f(x_k + \Delta_k \hat{d}_k) - f(x_k) + \rho(\Delta_k).$$

Putting the last two relations together, dividing through by Δ_k, and subtracting $\nabla f(x_k)^T \hat{d}_k$ from both sides yields

$$-\nabla f(x_k)^T \hat{d}_k \leq \left(\nabla f(x_k + \alpha_k \Delta_k \hat{d}_k) - \nabla f(x_k)\right)^T \hat{d}_k + \rho(\Delta_k)/\Delta_k.$$

From this we obtain

$$\kappa(\mathcal{G}_k) \parallel \nabla f(x_k) \parallel \parallel \hat{d}_k \parallel \leq \left(\nabla f(x_k + \alpha_k \Delta_k \hat{d}_k) - \nabla f(x_k)\right)^T \hat{d}_k + \rho(\Delta_k)/\Delta_k,$$

whence

$$\begin{aligned} \kappa(\mathcal{G}_k) \parallel \nabla f(x_k) \parallel &\leq \parallel \nabla f(x_k + \alpha_k \Delta_k \hat{d}_k) - \nabla f(x_k) \parallel + \rho(\Delta_k)/(\Delta_k \parallel \hat{d}_k \parallel) \\ &\leq M \Delta_k \parallel \hat{d}_k \parallel + \rho(\Delta_k)/(\Delta_k \parallel \hat{d}_k \parallel), \end{aligned}$$

and the result follows from (3.4). □

A similar bound can be obtained assuming only continuous differentiability of f. Let ω denote the following modulus of continuity of $\nabla f(x)$; i.e., given $r > 0$,

$$\omega(x, r) = \max \{ \parallel \nabla f(y) - \nabla f(x) \parallel \mid \parallel y - x \parallel \leq r \}.$$

Then the proof of Theorem 3.3 yields the bound

$$\parallel \nabla f(x_k) \parallel \leq \kappa(\mathcal{G}_k)^{-1} \left[\omega(x_k, \Delta_k \beta_{\max}) + \frac{\rho(\Delta_k)}{\Delta_k \beta_{\min}} \right].$$

3.7. Globalization Strategies in GSS. In light of Theorem 3.3, the globalization strategies for GSS may be viewed as techniques that ensure

$$\lim_{\substack{k \to +\infty \\ k \in \mathcal{K} \subseteq U}} \Delta_k = 0. \tag{3.13}$$

Recall that $\mathcal{U}$ is the set of *unsuccessful* iterations.

With time and imagination, one could concoct an endless variety of GSS algorithms for which (3.13) holds. In fact, that is part of the charm of GSS methods. Here we present three globalization strategies to ensure (3.13):

1. Sufficient decrease, used by Yu [278], Lucidi and Sciandrone [174], and García-Palomares and Rodríguez [115], is discussed in section 3.7.1.
2. Rational lattices, used by Berman [21, 22], Céa [56], Polak [211], Torczon [257], and Lewis and Torczon [166], are discussed in section 3.7.2.
3. Moving grids, which were motivated by the work of Coope and Price [77], are discussed in section 3.7.3.

A few comments are in order concerning the relationship between the globalization strategies used in GSS methods and the globalization strategies used in line search methods. As discussed in section 2.2, having a direction of descent at each iteration is not enough to guarantee convergence. One must also safeguard the *length* of the step. Recall the step-length pathology that must be prevented: taking too few steps of the appropriate length, relative to the amount of improvement that otherwise could be realized.

First consider the problem of preventing steps that are too short. As noted in section 2.2, it is well known in line search methods for gradient-based optimization that backtracking eliminates this problem [192, 197]. But GSS algorithms are, in a sense, doing a backtracking line search along the directions in $\mathcal{D}_k$. At each unsuccessful iteration—and only at unsuccessful iterations—GSS methods reduce the step-length control parameter and try again. This ensures that GSS methods take shorter steps only when absolutely necessary to make progress, providing a built-in "derivative-free" safeguard for GSS that has the same effect as enforcing (2.5).

Next consider the problem of preventing steps that are too long. For a conventional gradient-based line search method, this is accomplished by enforcing a sufficient decrease condition such as (2.4). However, GSS methods do not explicitly compute or estimate derivatives, so a step acceptance criterion such as (2.4) is not appropriate.

Failure due to infinitely many steps that are too long is pathological, and this possibility is prevented within GSS methods by simple techniques that do not require knowledge of derivatives. In connection with GSS, we discuss three mechanisms for avoiding steps that are too long. The first is a sufficient decrease condition that does not require derivatives. The other globalization techniques, in the rational lattice and moving grid approaches, prevent steps that are too long by restricting the types of steps that can be taken.

3.7.1. Globalization via the Sufficient Decrease Condition. Of the globalization approaches mentioned in section 3.7, sufficient decrease will be most familiar to those who have seen derivative-based methods. The earliest reference we have found to a sufficient decrease condition for a GSS method is by Yu [278]. Derivative-free sufficient decrease conditions for line search methods were later introduced by De Leone, Gaudioso, and Grippo [87]. The ideas in [87], as well as the further developments in [127], were adapted by Lucidi and Sciandrone [174] within the context of direct search methods. Shortly afterward, similar ideas were proposed by García-Palomares and Rodríguez [115].

The sufficient decrease criterion says that GSS methods accept a new iterate only if it satisfies $f(x_k + \Delta_k d_k) < f(x_k) - \rho(\Delta_k)$ where the forcing function $\rho(t)$ has the properties given in (3.9).

It is worth comparing this step acceptance condition with the the first of the Armijo–Goldstein–Wolfe conditions (2.4). Define

$$g(\Delta) = f(x_k + \Delta d_k) - f(x_k) \quad \text{and} \quad h(\Delta) = -\rho(\Delta).$$

Observe that $g(0) = h(0) = 0$. If f is continuously differentiable and d is a descent direction, then $g'(0) < 0$. Because of assumption (3.9), it follows that $h'(0) = 0$. Thus, the situation illustrated in Figure 3.4, which is analogous to the situation illustrated in Figure 2.2, is that there exists a range of Δ values for which $g(\Delta) < h(\Delta)$ and, consequently,

$$f(x_k + \Delta d_k) - f(x_k) < -\rho(\Delta).$$

To show that the sufficient decrease condition forces a subsequence of Δ_k's to 0, the only requirement is that f be bounded below. The theorem and its proof are very similar to results for line search methods.

THEOREM 3.4. *Let f be bounded below. Suppose that $\rho(t)$ is as specified in* (3.9). *Then GSS produces iterations satisfying*

$$\liminf_{k \to +\infty} \Delta_k = 0.$$

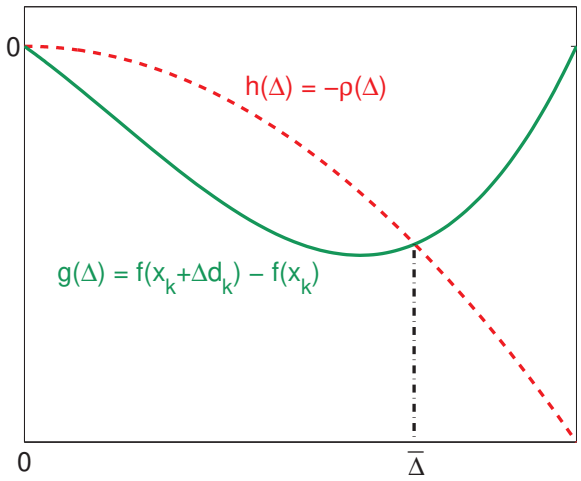


Fig. 3.4 *Suitably small step lengths will satisfy the sufficient decrease condition.*

Proof. Suppose not. Then there exists $\Delta_* > 0$ such that $\Delta_k \geq \Delta_*$ for all k. The updating rule for Δ_k, given in (3.6), implies that the number of successful iterations must be infinite; i.e., $\mathcal{S}$ is infinite. From the assumption in (3.9) that $\rho(\cdot)$ is a nondecreasing function and the lower bound on Δ_k, there exists $\rho_* > 0$ such that $\rho_* \leq \rho(\Delta_k)$ for all k. Now, for every $k \in \mathcal{S}$,

$$f(x_{k+1}) - f(x_k) \leq -\rho(\Delta_k) \leq -\rho_* < 0,$$

while for all other iterations (i.e., $k \in \mathcal{U}$), $f(x_{k+1}) - f(x_k) = 0$. Thus, $f(x_k) \to -\infty$, which contradicts the assumption that f is bounded below. Hence, the claim. □

This lemma can be furthered improved to show that the entire sequence of Δ_k's goes to zero by assuming that the expansion parameter ϕ_k is bounded above.

The methods proposed by Lucidi and Sciandrone [174] and García-Palomares and Rodríguez [115], which have motivated the current discussion of a derivative-free sufficient decrease condition, include additional interesting algorithmic features. These are discussed in section 5.

For the Connoisseur. Note that Theorem 3.4 depended on the updating rule for Δ_k in (3.6). Specifically, if there were only finitely many successful iterations, the Δ_k's would be forced to go to zero since every contraction satisfies

$$0 < \theta_k \leq \theta_{\max} < 1.$$

The upper bound on the contraction parameter plays a critical role. If this requirement were omitted, the following pathology could arise, where $\theta_k < 1$ is always chosen but Δ_k is never reduced to less than $\frac{1}{2}\Delta_0$.

Suppose $x_0 = x_* = \min f(x)$, so that every GSS iteration is unsuccessful. For each iteration k, define the contraction parameter θ as

$$\theta_k = \frac{k+3}{k+2} \cdot \frac{k+1}{k+2} = \frac{k^2+4k+3}{k^2+4k+4} \in (0,1).$$

Recall that at unsuccessful iterations, θ is used in the update $\Delta_{k+1} = \theta_k \Delta_k$ (see (3.6)). In this example, every iteration is unsuccessful, so

$$\Delta_k = \left(\prod_{i=1}^{k} \theta_i \right) \Delta_0.$$

Observe that

$$\prod_{i=1}^{k} \theta_i = \left(\frac{k+3}{k+2} \cdot \frac{k+1}{k+2} \right) \left(\frac{k+2}{k+1} \cdot \frac{k}{k+1} \right) \cdots \left(\frac{4}{3} \cdot \frac{2}{3} \right) \left(\frac{3}{2} \cdot \frac{1}{2} \right) = \frac{1}{2} \left(\frac{k+3}{k+2} \right).$$

Therefore,

$$\lim_{k \to \infty} \Delta_k = \lim_{k \to \infty} \frac{1}{2} \left(\frac{k+3}{k+2} \right) \Delta_0 = \frac{1}{2} \Delta_0.$$

In other words, the step length will never be less than $\frac{1}{2}\Delta_0$.

3.7.2. Globalization via a Rational Lattice. GSS algorithms also can accept steps that yield only simple decrease. In this case, convergence can be ensured by restricting the types of steps GSS is allowed to take. The lattice-based methods discussed in this section, and the grid-based methods discussed in the next section, restrict iterates to lie on meshes that are controlled implicitly. The earliest observations of which we are aware on the connection between simple decrease and the use of lattices as a step-length control mechanism are due to Berman [22] and Polak [211]. The lattice approach as a general globalization technique was later formalized by Torczon in [257].

The basic idea is quite simple. Consider compass search. The choice of step lengths and directions means that the iterates lie on an ever-shrinking mesh of fixed orientation (actually, uniform Cartesian grids). (We discuss lattices whose orientations are not fixed, which we call moving grids, in section 3.7.3.) The spacing of the mesh elements depends on Δ_k, so the lattice is refined only when Δ_k is reduced. The additional structure these features provide suffices to prevent the pathological steps of Figure 2.1(a).

Imposing some minor requirements on the search directions and the update rules for the step-length control parameter ensures that every iterate lies on a rational lattice. From this fact, together with the simple decrease criterion for accepting steps, it will follow that (3.13) holds; i.e., some subsequence of the step lengths goes to zero.

First consider the structure of the step-length control parameter under two simple conditions:

$$\theta_k = \Lambda^{m_k}, \quad m_k \in \{-1, -2, \ldots\}, \text{ for all } k \in \mathcal{U}, \tag{3.14}$$

$$\phi_k = \Lambda^{\ell_k}, \quad \ell_k \in \{0, 1, 2, \ldots\}, \text{ for all } k \in \mathcal{S}, \tag{3.15}$$

where $\Lambda > 1$ is an integer that does not change with k. In other words, the contraction factor is a negative integral power of Λ and the expansion factor is a nonnegative integral power of Λ. For simplicity, it helps to assume that all choices look something like

$$\theta_k = \frac{1}{2} \text{ for all } k \in \mathcal{U}, \quad \phi_k = 1 \text{ for all } k \in \mathcal{S}.$$

These were the values used in the description of compass search in section 1.1 and have been the conventional choice for decades.

Using (3.14) and (3.15), the following lemma is straightforward.

LEMMA 3.5. *In GSS, updating* Δ_k *according to* (3.6) *with contraction and expansion parameters satisfying* (3.14) *and* (3.15),

$$\Delta_{k+1} = \Lambda^{\Gamma_k}\Delta_0 \text{ with } \Gamma_k \in \mathbb{Z}. \tag{3.16}$$

Next, consider the search directions. Each generating set $\mathcal{G}_k$ should be drawn from a finite set; i.e.,

$$\mathcal{G}_k \subseteq \mathbf{G} \quad \text{for all } k = 1, 2, \ldots, \tag{3.17}$$

where

$$\mathbf{G} = \{d^{(1)}, \ldots, d^{(p)}\}. \tag{3.18}$$

An incidental consequence of this assumption is that the lower bound $\kappa_{\min} > 0$ is implicit. Because the number of possible subsets of $\mathbf{G}$ is finite, $\kappa_{\min}$ is the least cosine measure of any subset that is also a generating set.

Structure is also imposed on the set $\mathcal{H}_k$. It consists of vectors that are nonnegative integer combinations of the vectors in $\mathbf{G}$:

$$\mathcal{H}_k \subset \left\{ \sum_{i=0}^{p} \xi^{(i)} d^{(i)} \mid \xi^{(i)} \in \{0, 1, 2, \ldots\} \right\}. \tag{3.19}$$

Now consider the update of x_k at the end of each iteration k. If k is unsuccessful (i.e., $k \in \mathcal{U}$), then $x_{k+1} = x_k$. If k is successful (i.e., $k \in \mathcal{S}$), then $x_{k+1} = x_k + \Delta_k d_k$ for some $d_k \in \mathcal{D}_k$. This leads to the following observation.

LEMMA 3.6. *In GSS, updating* x_k *according to* (3.5) *and assuming each* $\mathcal{G}_k$ *satisfies* (3.17) *and each* $\mathcal{H}_k$ *satisfies* (3.19), *there exist nonnegative integers* $\alpha_k^{(i)}$ *for each* $i = 1, \ldots, p$ *and* $k = 0, 1, 2, \ldots$ *such that*

$$x_{k+1} = x_0 + \sum_{i=1}^{p} \delta_k(i)\, d^{(i)}, \quad \text{where} \quad \delta_k(i) = \sum_{j=0}^{k} \alpha_j^{(i)} \Delta_j. \tag{3.20}$$

Proof. The proof is by induction. For $k = 0$, there are three possible outcomes:

1. $k \in \mathcal{U}$ (i.e., the step is unsuccessful),
2. $k \in \mathcal{S}$ and $d_k \in \mathcal{G}_k$ (i.e., the step is successful and (3.17) and (3.18) hold), and
3. $k \in \mathcal{S}$ and $d_k \in \mathcal{H}_k$ (i.e., the step is successful and (3.19) holds).

In case 1, (3.5) says that $x_1 = x_0$, so simply choose $\alpha_0^{(i)} = 0$, so that $\delta_0(i) = 0$ for $i = 1, \ldots, p$. In case 2, (3.5) says that $x_1 = x_0 + \Delta_0 d_0$; (3.17) and (3.18) say that $d_0 = d^{(\ell)} \in \mathbf{G}$ for some $\ell \in \{1, \ldots, p\}$. So choose $\alpha_0^{(\ell)} = 1$, so that $\delta_0(\ell) = \Delta_0$, and $\alpha_0^{(i)} = 0$, so that $\delta_0(i) = 0$ for all $i \in \{1, \ldots, p\} \setminus \{\ell\}$. In case 3, again (3.5) yields $x_1 = x_0 + \Delta_0 d_0$; (3.19) says that there exist $\xi^{(i)}$ such that $d_0 = \sum_{i=1}^{p} \xi^{(i)} d^{(i)}$. Choose $\alpha_0^{(i)} = \xi^{(i)}$, so that $\delta_0(i) = \Delta_0 \xi^{(i)}$ for $i = 1, \ldots, p$.

Now assume that (3.20) holds for x_k and consider x_{k+1}. Again there are three possible outcomes. If $k \in \mathcal{U}$, then choose $\alpha_k^{(i)} = 0$, so that $\delta_k(i) = \delta_{k-1}(i) = \sum_{j=0}^{k-1} \alpha_j^{(i)} \Delta_j$

for all $i \in \{1, \ldots, p\}$. On the other hand, if $k \in \mathcal{S}$ and $d_k \in \mathcal{G}_k$, so that $d_k = d^{(\ell)} \in \mathbf{G}$ for some $\ell \in \{1, \ldots, p\}$, then choose $\alpha_k^{(\ell)} = 1$, so that $\delta_k(\ell) = \Delta_k + \sum_{j=0}^{k-1} \alpha_j^{(\ell)} \Delta_j$, and $\alpha_k^{(i)} = 0$, so that $\delta_k(i) = \delta_{k-1}(i) = \sum_{j=0}^{k-1} \alpha_j^{(i)} \Delta_j$ for all $i \in \{1, \ldots, p\} \setminus \{\ell\}$. Finally, if $k \in \mathcal{S}$ and $d_k \in \mathcal{H}_k$, then (3.19) says there exist $\xi^{(i)}$ such that $d_k = \sum_{i=1}^{p} \xi^{(i)} d^{(i)}$. Choose $\alpha_k^{(i)} = \xi^{(i)}$, so that $\delta_k(i) = \Delta_k \xi^{(i)} + \sum_{j=0}^{k-1} \alpha_j^{(i)} \Delta_j$ for $i = 1, \ldots, p$. Hence, the claim. □

So, there are two pieces here—we know something about the structure of each Δ_k and something about the structure of each x_k. One more assumption is needed to show that every iterate lies on a lattice. Assume that each search direction $d \in \mathbf{G}$ is integral, i.e.,

$$d \in \mathbb{Z}^n \text{ for each } d \in \mathbf{G}. \tag{3.21}$$

Combining Lemmas 3.5 and 3.6 with (3.21) leads to the following theorem from [257].

THEOREM 3.7. *Suppose that conditions* (3.14), (3.15), (3.17), (3.19), *and* (3.21) *hold. Let* Γ_k *be defined as in Lemma* 3.5, *and let* $\Gamma > 0$ *be a constant. Then GSS produces iterates such that for any* k *with*

$$\Gamma \leq \Gamma_j \text{ for all } j \leq k,$$

the following holds:

$$x_{k+1} = x_0 + \Lambda^{\Gamma}\, \Delta_0 \sum_{i=1}^{p} \zeta_k(i, \Gamma)\, d^{(i)}, \tag{3.22}$$

where

$$\zeta_k(i, \Gamma) = \sum_{j=0}^{k} \alpha_j^{(i)} \Lambda^{\Gamma_j - \Gamma} \in \mathbb{Z}, \tag{3.23}$$

and $\alpha_j^{(i)}$, *for each* $i = 1, \ldots, p$ *and* $k = 0, 1, 2, \ldots$, *is defined in Lemma* 3.6.

The significance of this result is that, since $\zeta_k(i, \Gamma)$ and $d^{(i)}$ are integral, x_k lies on an integral lattice scaled by $\Lambda^{\Gamma}\, \Delta_0$ and translated by x_0; we denote the lattice by

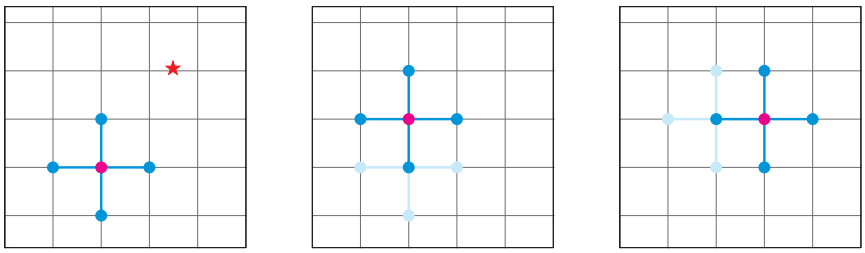
$$\mathcal{M}(x_0, \Delta_0, \mathbf{G}, \Lambda, \Gamma) = \left\{ x_0 + \Lambda^{\Gamma} \Delta_0 \sum_{i=1}^{p} \zeta^{(i)} d^{(i)} \mid \zeta^{(i)} \in \mathbb{Z} \right\}. \tag{3.24}$$

Consider again the example of compass search applied to the two-dimensional function in Figure 1.1. That example is revisited in Figure 3.5, but this time the corresponding lattices rather than the level curves of the objective function are shown. Figure 3.6 depicts the lattice at iteration $k = 5$. Here $\Gamma = -1$ defines the scale of the lattice. The other parameters that define the lattice are

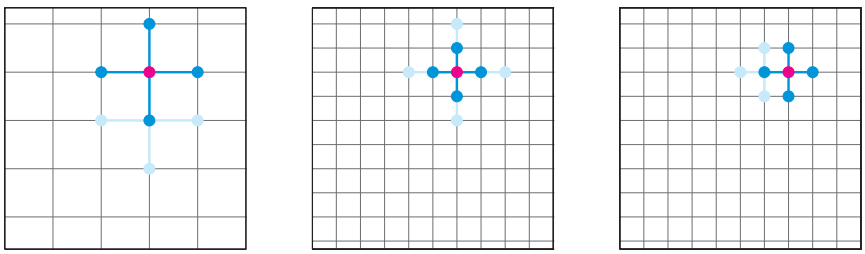
$$x_0 = \begin{bmatrix} -0.9 \\ -1.0 \end{bmatrix}, \quad \Delta_0 = 0.3, \quad \mathbf{G} = \{e_1, -e_1, e_2, -e_2\}, \quad \Lambda = 2.$$

As can be seen from the figure,

$$x_5 = x_0 + 2^{-1} \Delta_0 \left(3d^{(1)} + 4d^{(3)} \right).$$



(a) $k = 0$ (b) $k = 1$ (c) $k = 2$



(d) $k = 3$ (e) $k = 4$ (Refine!) (f) $k = 5$

Fig. 3.5 *Illustrating the underlying lattice.*

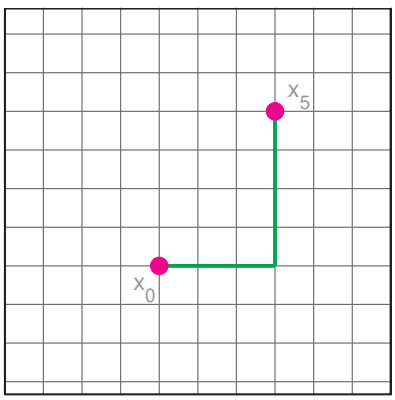


Fig. 3.6 *All iterates lie on a rational lattice.*

Thus, to reach x_5 from x_0, take seven "hops" across the lattice $\mathcal{M}(x_0, \Delta_0, \mathbf{G}, \Lambda, \Gamma)$: three to the East and four to the North. This means that the values for the $\zeta_k(i, \Gamma)$ in (3.23) are

$$\zeta_4(1, -1) = 3, \quad \zeta_4(2, -1) = 0, \quad \zeta_4(3, -1) = 4, \quad \zeta_4(4, -1) = 0.$$

To appreciate the role played by $\mathcal{H}_k$, assume the same conclusion shown in Figure 3.6, but under a scenario in which the search arrives at the point marked x_5 in one

step $\Delta_0 d_0$ of the algorithm (i.e., within a single iteration). In this situation, $\Gamma = 0$ defines the scale of the lattice. The other parameters that define the lattice are given by

$$x_0 = \begin{bmatrix} -0.9 \\ -1.0 \end{bmatrix}, \quad \Delta_0 = 0.15, \quad \mathbf{G} = \{e_1, -e_1, e_2, -e_2\}, \quad \Lambda = 2.$$

(Note that in the previous example $\Gamma = -1$ and $\Delta_0 = 0.3$, so that while now $\Gamma = 0$ and $\Delta_0 = 0.15$, $\mathcal{M}(x_0, \Delta_0, \mathbf{G}, \Lambda, \Gamma)$ still "looks" the same since x_0, $\mathbf{G}$, and Λ are unchanged and $\Lambda^\Gamma \Delta_0 = 0.15$ in either instance.)

Let $\mathcal{G}_0 \equiv \mathbf{G}$. Let $\mathcal{H}_0 = \{(3, 4)^T\}$, which satisfies (3.19) with $\xi^{(1)} = 3$, $\xi^{(3)} = 4$, and $\xi^{(2)} = \xi^{(4)} = 0$. Suppose the search starts with the step $\Delta_0 d$, $d \in \mathcal{H}_0$. Then $k = 0 \in \mathcal{S}$ (the simple decrease condition is satisfied by this choice of d) and

$$x_1 = x_0 + 2^0 \Delta_0 \left(3d^{(1)} + 4d^{(3)} \right).$$

In other words, this is still seven "hops" across the lattice $\mathcal{M}(x_0, \Delta_0, \mathbf{G}, \Lambda, \Gamma)$: three to the East and four to the North. Now, however,

$$\zeta_0(1, 0) = 3, \quad \zeta_0(2, 0) = 0, \quad \zeta_0(3, 0) = 4, \quad \zeta_0(4, 0) = 0;$$

it took only one iteration, rather than four iterations, to reach this same lattice point.

Once $\mathcal{H}_k$ is restricted as in (3.19), it plays only a minor role in the analysis. But if one has heuristics that suggest particularly good directions (or lattice points) other than those defined by $\mathcal{G}_k$, one can consider steps in those directions first in an attempt to accelerate the search. Such algorithmic possibilities are described further in section 4. The important observation is that (3.19) ensures that the underlying structure of the iterates is unchanged.

To summarize, so long as the conditions that ensure a rational lattice are enforced, every iterate of a GSS method that uses a rational lattice globalization strategy can be described as in (3.22). Specifically, the iterates always are guaranteed to lie on a lattice if

1. the expansion and contraction parameters are restricted to integral powers of some fixed integer Λ;
2. all generating sets $\mathcal{G}_k$ are drawn from a finite set of search directions $\mathbf{G}$;
3. any additional search directions contained in $\mathcal{H}_k$ are nonnegative integer combinations of the directions in $\mathbf{G}$; and
4. each search direction in $\mathbf{G}$ is integral.

In general, for unconstrained methods, these conditions are simple to enforce. Some relaxation of these conditions are possible [257]. A few of these relaxations are described in section 5.

To show that there is a subsequence of Δ_k tending to zero (i.e., (3.13)), we make the additional assumption that the level set $\mathcal{L}_f(x_0)$ of f is bounded. Then the intersection of $\mathcal{L}_f(x_0)$ with any *fixed* lattice $\mathcal{M}$ is finite. This is useful for the following reason. Recall that the sequence of successful GSS iterates must strictly improve the function value; i.e., the sequence

$$\{f(x_k)\}_{k \in \mathcal{S}}$$

is strictly decreasing. This means that GSS can visit each point on the fixed lattice at most once, so there can be only a finite number of successful iterates for a lattice of fixed size.

A lattice of fixed size is important for the following reason. If no subsequence of the Δ_k's goes to zero, then the lattice is only refined to a certain point and not beyond; i.e., GSS is working on a fixed lattice. This is the key to the following result.

THEOREM 3.8. *Let the level set $\mathcal{L}_f(x_0)$ be bounded. Choose the expansion and contraction parameters according to the conditions* (3.14) *and* (3.15)*, and the set of search directions according to the conditions* (3.17)*,* (3.19)*, and* (3.21)*. Then GSS produces iterates such that*

$$\liminf_{k\to\infty} \Delta_k = 0.$$

Proof. Suppose not. Then there exists $\Delta_* > 0$ such that $\Delta_k > \Delta_*$ for all k. This bound on Δ_k in turn implies, by (3.14), (3.15), and Lemma 3.5, that there must exist a Γ_* such that $\Gamma_* \leq \Gamma_k$ for all k, where Γ_k is as defined in Lemma 3.5. With (3.17), (3.19), (3.21), and Theorem 3.7, this implies that *every* iterate x_k must lie on the lattice, $\mathcal{M}(x_0, \Delta_0, \mathbf{G}, \Lambda, \Gamma_*)$.

Observe that any point on the lattice corresponds to at most one successful iteration, according to the decrease condition (3.7). On the other hand, every iterate must lie in $\mathcal{L}_f(x_0)$, which by assumption is bounded. The intersection of $\mathcal{M}(x_0, \Delta_0, \mathbf{G}, \Lambda, \Gamma_*)$ and $\mathcal{L}_f(x_0)$ is therefore finite.

Since any successful iterate must be in the finite set $\mathcal{M}(x_0, \Delta_0, \mathbf{G}, \Lambda, \Gamma_*) \cap \mathcal{L}_f(x_0)$, and no iterate can be successful more than once, it follows that $\mathcal{S}$ must be finite.

If $\mathcal{S}$ is finite and, consequently, $\mathcal{U}$ is infinite, according to (3.6), Δ_k expands only finitely many times and contracts infinitely many times. Each contraction is by a factor of at least $\min\{\Lambda^{-1}, \theta_{\max}\}$, which guarantees that

$$\lim_{k\to\infty} \Delta_k = 0.$$

This is a contradiction and the claim follows. ∎

Note that the only assumption on f made in Theorem 3.8 is that its level set $\mathcal{L}_f(x_0)$ is compact. Observe also that Theorem 3.8 is independent of the choice of ρ and holds for the sufficient decrease condition as well.

For the Connoisseur. Before continuing, we give a brief discussion of why we refer to this as a *rational lattice* (as opposed to an *integer lattice*) globalization strategy. Note that the assumption that Λ and the search directions $d \in \mathbf{G}$ are integral can be relaxed as follows. Let $\Lambda \in \mathbb{Q}$ and let $\mathbf{G}$ be such that

$$d = Bc, \text{ where } c \in \mathbb{Q}^n \text{ for each } d \in \mathbf{G}. \tag{3.25}$$

Here $B \in \mathbb{R}^{n\times n}$ is a fixed nonsingular matrix.

Theorem 3.7 can be rewritten to incorporate these changes (see either [257] or [161] for further detail), but these relaxations do not fundamentally alter the conclusion of the theorem, nor the implication, which is that the lattice can be represented as in (3.24), possibly subject to some minor modifications.

Specifically, the relaxation of $\Lambda \in \mathbb{Q}$ changes the constant Λ^Γ in (3.22) and (3.24), but the $\zeta_k(i, \Gamma)$ in (3.23) remain integral, which is what is critical for the result. Further, the real matrix B is fixed, so it can be factored out of the summations in (3.22) and (3.24) along with Δ_0, which also may be real. The important term in each of (3.22) and (3.24) is the summation $\sum_{i=1}^p \zeta^{(i)} d^{(i)}$ (which becomes $B \sum_{i=1}^p \zeta^{(i)} c^{(i)}$ under the relaxation (3.25)) since it is the summation that yields the lattice generated by the vectors in the generating set $\mathbf{G}$. Since ultimately (3.23) and one of either (3.21)

or (3.25) hold, the summation, either $\sum_{i=1}^{p} \zeta^{(i)} d^{(i)}$ or $\sum_{i=1}^{p} \zeta^{(i)} c^{(i)}$, yields a rational lattice. Thus we call it a rational lattice globalization strategy, even though the lattice is translated by x_0, which is in $\mathbb{R}$, is scaled by Δ_0, which is in $\mathbb{R}$, and possibly may be both scaled and reoriented by B, which is in $\mathbb{R}^{n\times n}$. The fundamental structure upon which the analysis relies is the rational lattice $\sum_{i=1}^{p} \zeta^{(i)} d^{(i)}$ (or possibly $\sum_{i=1}^{p} \zeta^{(i)} c^{(i)}$), which is unchanged by these relaxations.

These relaxations were not incorporated into the preceding discussion because they complicate the presentation without providing further illumination of the final analytical results. In fact, the traditional choices for many of the classical algorithms have been $B = I$ (where I is the identity matrix) and $\Lambda = 2$. However, these relaxations are an important ingredient in the definition of some more recent methods. (See, for instance, either [256], and the further elaborations in [166], or the algorithmic development in [5].)

Audet [10] presents examples that demonstrate both why either d must be integral, or the c in (3.25) must be rational, and why the restrictions on the choice of contraction/expansion values are necessary.

3.7.3. Globalization via Moving Grids. The critical step in the proof of Theorem 3.8 is the observation that, given a fixed lattice $\mathcal{M}(x_0, \Delta_0, \mathbf{G}, \Lambda, \Gamma_*)$ over a compact set $\mathcal{L}_f(x_0)$, the requirement of strict improvement in f means that there are only finitely many points that can be considered before Δ_k must contract.

Coope and Price [77] observed that when Δ_k contracts, the search can, to a certain extent, start over; i.e., rather than simply reducing the scale factor Λ, one can also change $\mathbf{G}$. (In fact, [77] also exploits this ability to start over in another way: one can move from the current iterate to *any* point with a lower function value using a "finite process"; this is discussed in section 5. Here we concentrate on changes to $\mathbf{G}$.)

The desire to adapt the set of search directions, as a strategy to improve the asymptotic performance of the search, motivates this change in perspective. After evaluating f over the current lattice $\mathcal{M}$, why not use the information accumulated during this search on $\mathcal{M}$ to construct a new set of generators that attempt to capture, at least locally, some curvature information? This is in the spirit of several of the early direct search methods such as Rosenbrock's method of rotating directions [227], Powell's method based on conjugate directions [213], and the adaptive simplex of Nelder and Mead [194]. The algorithmic strategies in the former two papers, in particular, informed the developments in [76, 77, 224].

The strategy is essentially to fix the lattice structure between any two *unsuccessful* iterations by choosing a fixed set of *grid generators* defined by $\mathcal{G}_k$. First, $\mathcal{G}_k$ must be a finite set, i.e.,

$$\mathcal{G}_k = \{d_k^{(1)}, \ldots, d_k^{(p_k)}\}. \tag{3.26}$$

Second, the set must be chosen in such a way that it actually generates a lattice. Thus, we require that

$$d = Bc, \text{ where } c \in \mathbb{Q}^n \text{ for each } d \in \mathcal{G}_k. \tag{3.27}$$

Here $B \in \mathbb{R}^{n\times n}$ is a fixed nonsingular matrix. This is the same as in (3.25). Finally, the grid cannot be modified and the step length cannot be reduced until an unsuccessful point is encountered:

$$\mathcal{G}_k = \mathcal{G}_{k-1} \quad \text{for all} \quad k \in \mathcal{S}. \tag{3.28}$$

The set $\mathcal{H}_k$ is restricted to be integer combinations of elements of $\mathcal{G}_k$:

$$\mathcal{H}_k \subset \left\{ \sum_{i=0}^{p_k} \xi_k^{(i)} d_k^{(i)} \mid \xi_k^{(i)} \in \{0, 1, 2, \ldots\} \right\}. \tag{3.29}$$

The goal is to use the information accumulated during the successful iterations to construct a new—possibly better—set of search directions. The conditions on an iteration being unsuccessful guarantee a minimum of $n+1$ new objective values when the next set of grid generators is constructed.

For now we impose the requirement that $\phi_k = 1$ for all $k \in \mathcal{S}$. Given the GSS requirements for updating Δ_k, this means that $\Delta_k \leq \Delta_{k-1}$ for all k. "Long" steps are still possible given the definition of $\mathcal{H}_k$ in (3.29); in section 4.2 we demonstrate this possibility using the original pattern search algorithm of Hooke and Jeeves. The condition on ϕ_k can be relaxed by requiring, instead, that (3.13) holds, as is done in [278] and [77], rather than proving that (3.13) holds, as we will do here. Further variations on controlling Δ_k, including the possibility of *increasing* Δ_k after an unsuccessful iteration, as is allowed in [77], are discussed in section 5.

In order to state theoretical results, it is helpful to establish notation to define the most recent unsuccessful iterate. Let

$$\omega(k) = \max \{\ell \leq k \mid \ell \in \mathcal{U}\}. \tag{3.30}$$

Then we can establish the following theorem, which is the analogue of Theorem 3.7 in the case of the rational lattice. The proof is left to the reader.

THEOREM 3.9. *Let $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$ be updated according to* (3.26)–(3.29)*, and let $\phi_k = 1$ for all k. Then GSS produces iterates such that for any $k \in \mathcal{S}$*

$$x_k = x_{\omega(k)} + \Delta_{\omega(k)} \sum_{i=1}^{p_{\omega(k)}} \zeta_k(i)\, d_{\omega(k)}^{(i)},$$

where $\omega(k)$ denotes the most recent unsuccessful iterate as in (3.30) *and $\zeta_k(i)$ is integral for all i.*

Using the same notation as in the rational lattice case for (3.22), iterate x_k lies on the lattice defined by

$$\mathcal{M}(x_{\omega(k)}, \Delta_{\omega(k)}, \mathcal{G}_{\omega(k)}, 1, 1) = \left\{ x_{\omega(k)} + \Delta_{\omega(k)} \sum_{i=1}^{p_{\omega(k)}} \zeta^{(i)} d_{\omega(k)}^{(i)} \mid \zeta^{(i)} \in \mathbb{Z} \right\}. \tag{3.31}$$

Following the same line of reasoning in Theorem 3.8, for the moving grids globalization strategy, the sequence Δ_k is forced to go to zero.

THEOREM 3.10. *Let the level set of f, $\mathcal{L}_f(x_0)$, be bounded. Let $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$ be updated according to* (3.26)–(3.29)*, and let $\phi_k = 1$ for all $k \in \mathcal{S}$. Then GSS produces iterates such that*

$$\lim_{k \to \infty} \Delta_k = 0.$$

Proof. Suppose not. By the assumption that $\phi_k = 1$ for all k, the sequence Δ_k is decreasing as $k \to +\infty$ and *strictly* decreasing for $k \in \mathcal{U}$. So if the conclusion of the theorem is false, then there exists $\Delta_* > 0$ such that $\Delta_k > \Delta_*$ for all k.

Combining this with the fact that according to (3.6), GSS must reduce Δ_k by at least a factor of $\theta_{\max}$, there are only finitely many reductions of Δ_k if $\Delta_k > \Delta_*$ for

all k. Hence there can be only finitely many unsuccessful steps, so $\mathcal{S}$ is necessarily infinite.

Let $\hat{k} = \max\{k \mid k \in \mathcal{U}\}$ denote the index of the last unsuccessful iterate. By Theorem 3.9, every iterate $k \geq \hat{k}$ must lie on the grid defined by $\mathcal{M}(x_{\hat{k}}, \Delta_{\hat{k}}, \mathcal{G}_{\hat{k}}, 1, 1)$ as in (3.31).

Observe that every point on the lattice can generate at most one successful iterate, i.e., $k \in \mathcal{S}$, according to the decrease condition (3.7). On the other hand, every iterate must lie in $\mathcal{L}_f(x_0)$, which is assumed to be bounded. The intersection of $\mathcal{M}(x_{\hat{k}}, \Delta_{\hat{k}}, \mathcal{G}_{\hat{k}}, 1, 1)$ and $\mathcal{L}_f(x_0)$ is therefore finite.

Since any successful point must be in the finite set $\mathcal{M}(x_{\hat{k}}, \Delta_{\hat{k}}, \mathcal{G}_{\hat{k}}, 1, 1) \cap \mathcal{L}_f(x_0)$, and no point can be successful more than once, it follows that $\mathcal{S}$ must be finite, a contradiction. □

As with the lattice result, it makes no difference whether the simple or the sufficient decrease condition is used to accept a new iterate.

3.8. Putting It All Together—Global Convergence Results. We have discussed three ways to ensure that at least a subsequence of iterations for which Δ_k converges to zero using either sufficient decrease (section 3.7.1), the rational lattice (section 3.7.2), or moving grids (section 3.7.3). The assumptions we imposed to show the existence of such a subsequence are summarized below.

- Sufficient decrease—Assumes only that f is bounded below, and steps are accepted according to the sufficient decrease condition (3.7), where $\rho(\cdot)$ satisfies (3.9).
- Rational lattice—Assumes that the level set of f, $\mathcal{L}_f(x_0)$, is bounded. Further assumes that the expansion parameter ϕ_k and contraction parameter θ_k are restricted to be integral powers of some integral (or rational) constant Λ, as in (3.14) and (3.15). Finally, assumes the set of all search directions, $\mathbf{G} = \bigcup \mathcal{D}_k$ is finite and integral (or rational), as in (3.17) and (3.21).
- Moving grids—Assumes that the level set of f, $\mathcal{L}_f(x_0)$, is bounded. Further, assumes that the grid generators may only be updated after an unsuccessful iterations according to (3.26)–(3.29). Finally, assumes that no expansion steps are allowed (i.e., $\phi_k = 1$ for all $k \in \mathcal{S}$).

Recall, also, some of the requirements made in the specification of GSS given in Algorithm 3.2. All the directions in $\mathcal{D}_k$ are required to be bounded below in length. Furthermore, the directions in $\mathcal{G}_k$ are required to be bounded above in length. Each set of generators $\mathcal{G}_k$ requires a cosine measure $\kappa(\mathcal{G}_k)$ that is bounded below by $\kappa_{\min} > 0$. There is an upper bound $\theta_{\max} < 1$ on the contraction parameter. The latter requirement comes into play for globalization methods in sections 3.7.1–3.7.3, though only implicitly in the rational lattice discussion. The requirements on the forcing function come into play in the discussion of sufficient decrease in section 3.7.1.

3.8.1. The General Result. The following theorem is a synthesis of a variety of convergence results that have appeared in [77, 166, 174, 257, 278]. For consistency with the earlier discussion, we assume that f is continuously differentiable and that ∇f is Lipschitz continuous; however, one can prove the result assuming only that f is continuously differentiable.

THEOREM 3.11. *Let f be continuously differentiable on $\mathbb{R}^n$ with $\nabla f(x)$ Lipschitz continuous on $\mathbb{R}^n$ with constant M. Assume the level set of f, $\mathcal{L}_f(x_0)$, is bounded. Furthermore, assume that the requirements for one of the three globalization strategies hold; i.e., either*

1. *f is bounded below, and the sufficient decrease condition* (3.9) *holds;*
2. *the conditions* (3.14), (3.15), (3.17), (3.21) *hold to produce a rational lattice; or*
3. *the conditions* (3.26)–(3.29) *hold to produce a moving grid.*

Then GSS produces iterates such that

$$\liminf_{k\to+\infty} \| \nabla f(x_k) \| = 0.$$

The result follows from Theorem 3.3 and the fact that (3.13) holds via globalization.

3.8.2. A Stronger Result. Under some slightly stronger conditions, every limit point of the sequence of iterates produced by a GSS algorithm will be a stationary point. Specifically, we assume the following.

ASSUMPTION 3.12.

- *For any $k \in \mathcal{S}$, $f(x_k + \Delta_k d_k) \le f(x_k + \Delta_k d)$ for all $d \in \mathcal{G}_k$.*
- $\lim_{k\to+\infty} \Delta_k = 0$.

The first of these conditions says that GSS may only accept a step if it produces at least as much improvement as the *best* step among the search directions in $\mathcal{G}_k$. Practically, this means that the the objective must be evaluated at every trial step $x_k + \Delta_k d$ defined by every direction $d \in \mathcal{G}_k$.

When Assumption 3.12 holds, and Δ_k is small relative to $\| \nabla f(x_k) \|$, then the steps automatically satisfy a condition analogous to (2.4), as the next proposition shows.

PROPOSITION 3.13. *Suppose that Assumption* 3.12 *holds. Let f be continuously differentiable on $\mathbb{R}^n$ with $\nabla f(x)$ Lipschitz continuous on $\mathbb{R}^n$ with constant M. Then GSS produces iterates such that for any $\eta > 0$ there exist $\delta > 0$ and $\sigma > 0$, independent of k, such that if $\| \nabla f(x_k) \| > \eta$ and $\Delta_k < \delta$, then*

$$f(x_{k+1}) \le f(x_k) - \sigma\Delta_k \| \nabla f(x_k) \|.$$

Proof. Let $\hat{d}_k \in \mathcal{G}_k \subset \mathcal{D}_k$ satisfy (3.11); such a $\hat{d}_k$ exists because $\mathcal{G}_k$ generates $\mathbb{R}^n$. By the mean value theorem, for some $\alpha_k \in [0, 1]$,

$$f(x_k + \Delta_k\hat{d}_k) = f(x_k) + \Delta_k \nabla f(x_k)^T \hat{d}_k + \Delta_k \left[\nabla f(x_k + \alpha_k\Delta_k\hat{d}_k) - \nabla f(x_k) \right]^T \hat{d}_k.$$

Since $\hat{d}_k$ satisfies (3.11),

$$\begin{aligned} f(x_k + \Delta_k\hat{d}_k) - f(x_k) &\le \Delta_k \nabla f(x_k)^T \hat{d}_k + M\Delta_k^2 \| \hat{d}_k \|^2 \\ &\le -\kappa(\mathcal{G}_k)\Delta_k \| \nabla f(x_k) \| \, \| \hat{d}_k \| + M\Delta_k^2 \| \hat{d}_k \|^2. \end{aligned}$$

Define

$$\delta = \frac{\kappa_{\min}\eta}{2M\beta_{\max}}.$$

Then $\Delta_k < \delta$ and $\| \nabla f(x_k) \| > \eta$ imply

$$\begin{aligned} M\Delta_k^2 \|\hat{d}_k\|^2 &\le M\Delta_k \|\hat{d}_k\|^2 \left(\frac{\kappa_{\min}\eta}{2M\beta_{\max}} \right) \\ &\le \frac{1}{2}\kappa_{\min}\Delta_k\eta\|\hat{d}_k\| \\ &\le \frac{1}{2}\kappa(\mathcal{G}_k)\Delta_k\|\nabla f(x_k)\|\|\hat{d}_k\|. \end{aligned}$$

Thus,

$$f(x_k + \Delta_k \hat{d}_k) - f(x_k) \leq -\frac{1}{2}\kappa(\mathcal{G}_k)\Delta_k \,\|\, \nabla f(x_k) \,\|\, \|\, \hat{d}_k \,\|.$$

Setting $\sigma = \beta_{\max}\kappa_{\min}/2$ produces the desired result. □

Using Assumption 3.12, Proposition 3.13, and an argument originally introduced for trust region methods by Thomas [252], it can be shown that every limit point of the sequence of iterates x_k is a stationary point of f. As with Theorem 3.3, the Lipschitz requirement is not strictly necessary; see the observations that follow Theorem 3.3 for the essence of the alternative argument.

THEOREM 3.14. *Suppose that Assumption* 3.12 *holds and that GSS satisfies the additional requirement that* $\| d \| \leq \beta_{\max}$ *for all* $d \in \mathcal{H}_k$, $k = 1, 2, \ldots$, *where* $\beta_{\max}$ *is as defined in* (3.4). *Let* $\mathcal{L}_f(x_0)$ *be compact. Suppose* f *is continuously differentiable on* $\mathbb{R}^n$ *and* ∇f *is Lipschitz with constant* M. *Then for the sequence of iterates* $\{x_k\}$ *produced by the GSS algorithm,*

$$\lim_{k \to +\infty} \| \nabla f(x_k) \| = 0 \,.$$

That is, every limit point of the sequence of iterates is a stationary point.

Proof. Suppose not; then $\limsup_{k\to+\infty} \| \nabla f(x_k) \| = \varepsilon > 0$. Recall from Theorem 3.11 that $\liminf_{k\to+\infty} \| \nabla f(x_k) \| = 0$.

Then it is possible to define subsequences $\{k_i\}$ and $\{\ell_i\}$ such that

- $k_i < \ell_i$ for all $i = 1, 2, \ldots$,
- $\|\nabla f(x_{k_i})\| > \varepsilon/2$ for all $i = 1, 2, \ldots$,
- $\|\nabla f(x_{\ell_i})\| < \varepsilon/4$ for all $i = 1, 2, \ldots$, and
- $\|\nabla f(x_k)\| \geq \varepsilon/4$ for all $k_i \leq k < \ell_i$, $i = 1, 2, \ldots$.

Using $\eta = \varepsilon/4$ in Proposition 3.13 guarantees that for k sufficiently large (and Δ_k sufficiently small), there exists σ such that

$$f(x_k) - f(x_{k+1}) \geq \sigma\Delta_k \|\nabla f(x_k)\| \geq \varepsilon_1 \|x_{k+1} - x_k\|.$$

Here $\varepsilon_1 = (\varepsilon\sigma)/(4\beta_{\max})$.

Then

$$\varepsilon_1 \|x_{k_i} - x_{\ell_i}\| \leq \varepsilon_1 \sum_{k=k_i}^{\ell_i - 1} \|x_k - x_{k+1}\| \leq \varepsilon_1 \sum_{k=k_i}^{\ell_i - 1} f(x_k) - f(x_{k+1}) = \varepsilon_1 \left[f(x_{k_i}) - f(x_{\ell_i}) \right].$$

Now, since f is a strictly decreasing sequence and f is bounded below, it must converge. Thus, the right-hand side in the above equation must be going to zero, so $\|x_{k_i} - x_{\ell_i}\|$ must also be converging to zero as $i \to +\infty$. The assumption that ∇f is Lipschitz continuous means that for i sufficiently large we have

$$\|\nabla f(x_{k_i}) - \nabla f(x_{\ell_i})\| \leq \varepsilon/4.$$

From this follows the contradiction

$$\begin{aligned} \frac{\varepsilon}{2} &< \|\nabla f(x_{k_i})\| = \|(\nabla f(x_{k_i}) - \nabla f(x_{\ell_i})) + \nabla f(x_{\ell_i})\| \\ &\leq \|(\nabla f(x_{k_i}) - \nabla f(x_{\ell_i}))\| + \|\nabla f(x_{\ell_i})\| \\ &< \frac{\varepsilon}{4} + \frac{\varepsilon}{4} = \frac{\varepsilon}{2}. \end{aligned}$$

Hence, the theorem is proved. □

3.9. Some Comments on the Convergence of GSS under Simple Decrease. If only simple decrease in the objective is required to accept a step, and we do not require $\lim_{k\to\infty} \Delta_k = 0$, then we obtain the convergence result

$$\liminf_{k\to\infty} \| \nabla f(x_k) \| = 0 \tag{3.32}$$

for the sequence of iterates produced by a GSS method. Can one strengthen this result to be

$$\lim_{k\to\infty} \| \nabla f(x_k) \| = 0 \tag{3.33}$$

for a GSS method under the same hypotheses? The answer turns out to be no, in general. In [10], Audet constructs an example for which (3.32) holds but (3.33) does not. In this example, the sequence of iterates has an infinite number of accumulation points, one of which is not a stationary point, though the others are.

It is interesting to note that there is a similar situation for global convergence results for trust region algorithms. In [216], Powell allows steps to be accepted if they yield simple decrease in the objective, rather than the usual sufficient decrease condition for trust region algorithms. However, at such steps the radius of the trust region must be decreased. For such methods Powell shows that

$$\liminf_{k\to\infty} \| \nabla f(x_k) \| = 0.$$

Yuan [275] constructed an example that shows this result cannot be strengthened to

$$\lim_{k\to\infty} \| \nabla f(x_k) \| = 0.$$

A close examination of the mechanics of the convergence proofs for GSS and trust region methods, both with only simple decrease, reveals that they share a similar feature. In both, a sufficient decrease condition is satisfied implicitly. The step-length control mechanisms ensure that smaller steps are taken, but not too small, and once the steps are small enough, improvement is found, and the amount of decrease in the objective is related to the size of the gradient (see Proposition 3.13). Usually, this latter condition is explicitly enforced in gradient-based methods via the step acceptance criteria in line search methods and trust region methods.

3.10. Stopping Criteria. The global convergence results are asymptotic: they tell us what happens as $k \to +\infty$. As a practical matter, a good measure of success is needed to know when to terminate the search. Typically, unconstrained optimization methods terminate when the norm of the gradient is very small (among other tests). But without explicit gradient information in a GSS algorithm, how do we know when to stop?

Theorem 3.3 shows that at unsuccessful iterations, the norm of the gradient will tend to decrease as Δ_k is reduced. This means that it is reasonable to terminate the algorithm when Δ_k falls below some tolerance. This stopping criterion is, in fact, one that traditionally has been used [139, 194]. Theorem 3.3 shows the validity of this intuition of the original developers of direct search methods. Dolan, Lewis, and Torczon [98] give a further discussion of the role of Δ_k in stopping criteria and a numerical illustration of the relationship between $\nabla f(x_k)$ and Δ_k.

3.11. Local Convergence of GSS. GSS methods always include at least one descent direction in their set of search directions. Assuming we have a fairly good set of search directions (i.e., lots of them and well spread out so that the cosine measure is relatively large with respect to zero), then we may expect a GSS method to behave like steepest descent. This helps explains why GSS methods frequently get to the neighborhood of a solution quickly.

It is possible to show that under certain conditions, the entire sequence of iterates will converge to a single local minimizer and make statements about the rate of convergence for the *sequence of unsuccessful iterates*. First recall some standard definitions (e.g., [192, 197]). A sequence of scalars $\{\alpha_k\}$ is said to be *q-linearly convergent to zero* if there exists a constant $\beta \in (0, 1)$ such that

$$\frac{|\alpha_{k+1}|}{|\alpha_k|} \leq \beta \quad \text{for all } k \text{ sufficiently large.}$$

In the case of GSS, note that, if we require $\phi_k = 1$ for k sufficiently large, then the sequence $\{\Delta_k\}_{k\in\mathcal{U}}$ is q-linearly convergent to zero because $\Delta_{k_{i+1}}/\Delta_{k_i} \leq \theta_{\max}$.

A sequence $\{x_k\}$ is said to be *r-linearly convergent* to x_* if

$$\|x_k - x_*\| \leq \alpha_k \quad \text{for all } k \text{ sufficiently large,}$$

where $\{\alpha_k\}$ is a sequence of scalars that is q-linearly convergent to zero. We will show that under certain assumptions, which are typical of local convergence results for gradient-based methods, the sequence $\{x_k\}_{k\in\mathcal{U}}$ is r-linearly convergent to x_*, and, in fact, $\| x_k - x_* \|$ is $O(\Delta_k)$. Once again we see a kinship between GSS methods and steepest descent—for steepest descent with accurate line searches, one can prove local q-linear convergence to x_*.

Theorem 3.15 generalizes the local convergence result in [98] for methods that are based on simple decrease. Theorem 3.15 also encompasses methods that require sufficient decrease. The result says that if some iterate x_k lands in a "basin of attraction" near a local minimizer x_*, and the step lengths are appropriately small, then the entire sequence of iterates will converge to x_*.

The theorem is similar in spirit to local convergence results for algorithms that use finite-difference estimates of the gradient. For instance, the local convergence result in [29] requires that the points from whose objective values the finite-difference estimates of the gradients are computed must be sufficiently close to x_*. The requirement here that the entire pattern be close to x_* is similar.

THEOREM 3.15. *Let f be twice continuously differentiable. Suppose x_* is a local minimizer of f and that $\nabla^2 f(x_*)$ is positive definite. For GSS, assume the following.*

1. *$\phi_k = 1$ for all $k \in \mathcal{S}$;*
2. *$\rho(t) = \alpha t^p$ for some fixed $\alpha > 0$ and fixed $p \geq 2$; and*
3. *$\beta_{\min} \leq \|d\| \leq \beta_{\max}$ for all $d \in \mathcal{D}_k$ and all k.*

Then if x_0 is sufficiently close to x_*, Δ_0 is sufficiently small, and $\lim_{k\to\infty} \Delta_k = 0$, we are guaranteed that

$$\lim_{k\to+\infty} x_k = x_*.$$

Furthermore, for $k \in \mathcal{U}$, we have that

$$\|x_k - x_*\| \leq c\, \Delta_k$$

for some constant c independent of k. Thus, the sequence $\{x_k\}_{k\in\mathcal{U}}$ is r-linearly convergent.

Proof. Let $\phi_{\min}$ and $\phi_{\max}$ denote, respectively, the minimum and maximum eigenvalues of $\nabla^2 f(x_*)$.

Since we assume that $\nabla^2 f$ is continuous, there exists an $\eta > 0$ such that

$$x \in \mathcal{B}(x_*, \eta) \quad \Rightarrow \quad \text{spectrum}\left\{\nabla^2 f(x)\right\} \subset \left[\ \frac{1}{2}\,\phi_{\min}\ ,\ 2\,\phi_{\max}\ \right].$$

These bounds on the second derivative are used in several places in the proof.

First, for any $x_k \in \mathcal{B}(x_*, \eta)$, we have

$$\nabla f(x_k) - \nabla f(x_*) = \int_0^1 \left[\nabla^2 f(x_* + t(x_k - x_*))\ (x_k - x_*)\right]\ dt.$$

Since $\nabla f(x_*) = 0$, we have

$$\|\nabla f(x_k)\| = \left\| \int_0^1 \left[\nabla^2 f(x_* + t(x_k - x_*))\ (x_k - x_*)\right]\ dt \right\| \geq \frac{1}{2}\phi_{\min}\ \|x_k - x_*\|.$$

For any $k \in \mathcal{U}$, employing Theorem 3.3 yields

$$\|x_k - x_*\| \leq 2\frac{\|\nabla f(x_k)\|}{\phi_{\min}} \leq \frac{2}{\phi_{\min}\kappa_{\min}} \left(\frac{M}{2}\beta_{\max} + \frac{\alpha\Delta_0^{p-2}}{\beta_{\min}}\right) \Delta_k.$$

To summarize the first part of the argument,

$$\|x_k - x_*\| \leq c_1\Delta_k \quad \text{for any} \quad k \in \mathcal{U} \quad \text{such that} \quad x_k \in \mathcal{B}(x_*, \eta). \tag{3.34}$$

Second, let $x, y \in \mathcal{B}(x_*, \eta)$ such that $f(x) < f(y)$. From Taylor's theorem with remainder and the fact that $\nabla f(x_*) = 0$, we have

$$f(y) = f(x_*) + \frac{1}{2}(y - x_*)^T \nabla^2 f(\xi)(y - x_*),$$
$$f(x) = f(x_*) + \frac{1}{2}(x - x_*)^T \nabla^2 f(\omega)(x - x_*)$$

for ξ and ω on the line segments connecting x_* with y and x, respectively. Since $f(x) \leq f(y)$, we obtain

$$0 \leq f(y) - f(x) = \frac{1}{2}(y - x_*)^T \nabla^2 f(\xi)(y - x_*) - \frac{1}{2}(x - x_*)^T \nabla^2 f(\omega)(x - x_*),$$

whence

$$0 \leq 2\phi_{\max}\| y - x_* \|^2 - \frac{1}{2}\phi_{\min}\| x - x_* \|^2.$$

Summarizing the second part of the argument, there exists c_2 such that

$$\|x - x_*\| \leq c_2\ \|y - x_*\| \quad \text{for any} \quad x, y \in \mathcal{B}(x_*, \eta) \text{ with } f(x) \leq f(y). \tag{3.35}$$

For the third part of the argument, we must show that *all* the iterates stay in $\mathcal{B}(x_*, \eta)$. Assume that $x_0 \in \mathcal{B}(x_*, \eta)$ is close enough to x_* so that

$$\|x_0 - x_*\| \leq \frac{\eta}{2c_2},$$

where c_2 is from (3.35). Assume Δ_0 is small enough so that

$$\|\Delta_0\| \leq \frac{\eta}{2\beta_{\max}}.$$

Assuming that $x_k \in \mathcal{B}(x_*, \eta)$, we have

$$\|x_{k+1} - x_*\| \leq \|x_{k+1} - x_k\| + \|x_k - x_*\|.$$

From the assumptions on Δ_k, we have

$$\|x_{k+1} - x_k\| \leq \Delta_k \beta_{\max} \leq \Delta_0 \beta_{\max} \leq \frac{\eta}{2}.$$

Since $f(x_k) \neq f(x_0)$ and $x_0, x_k \in \mathcal{B}(x_*, \eta)$, we can apply (3.35) to obtain

$$\|x_k - x_*\| \leq c_2 \|x_0 - x_*\| \leq \frac{\eta}{2}.$$

Thus, the third element in the argument says

$$x_k \in \mathcal{B}(x_*, \eta) \quad \text{for all} \quad k = 1, 2, \ldots. \tag{3.36}$$

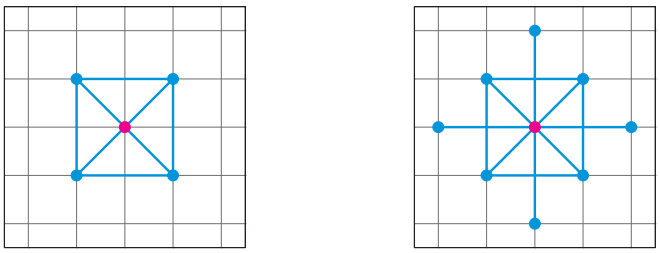
By assumption, $\lim_{k\to 0} \Delta_k = 0$. Furthermore, (3.36) says every iterate is in $\mathcal{B}(x_*, \eta)$. Let $k \in \mathcal{S}$ and let $\omega(k)$ denote the most recent unsuccessful iterate. Then by (3.35) and (3.34),

$$\|x_k - x_*\| \leq c_2 \|x_{\omega(k)} - x_*\| \leq c_1 c_2 \Delta_{\omega(k)}.$$

By the previous equation and (3.34), clearly $x_k \to x_*$. Furthermore, by (3.34) and (3.36), the unsuccessful iterates are r-linearly convergent. □

4. Algorithmic Options. The GSS framework is flexible enough to admit a wide range of algorithmic options. In particular, the choice of search directions is an important component when defining a GSS method. There is wide latitude in choosing the sets of search directions $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$ for a GSS method. The conditions on $\mathcal{G}_k$ are clear enough: the set of vectors must form a generating set. The set of coordinate directions $\mathcal{D}_\oplus$ has long been the traditional choice for $\mathcal{G}_k$. However, as early as 1960, Rosenbrock realized the potential algorithmic advantage of rotating a set of $2n$ orthogonal vectors [227] so that at least one of the directions more closely conformed to the local topography of the function. Much of the recent work on GSS methods has explored the possible algorithmic and computational advantages to be gained by choosing $\mathcal{G}_k \neq \mathcal{D}_\oplus$ [77, 78, 115, 140, 166, 174, 256].

The question we have not really addressed is the role of $\mathcal{H}_k$. Recall that the early development of direct search focused on heuristics. Compass search is an "obvious" algorithm and, as Davidon noted at the start, known to be "slow but sure." So the question in the late 1950s and early 1960s became what sort of heuristics to employ in an effort to accelerate the search. This is where $\mathcal{H}_k$ becomes valuable. The directions $\mathcal{H}_k$ contains play no substantive part in the analysis, but they allow the incorporation of heuristics chosen to improve the effectiveness of the search. With the development of a better understanding of what it takes to ensure convergence of GSS methods, the same question, what sort of heuristics to employ in an effort to accelerate the search, has reemerged in a new light.



(a) (b)

Fig. 4.1 *Examples in* $\mathbb{R}^2$ *of both* (a) *a two-level factorial design and* (b) *a composite design.*

4.1. Statistical Design of Experiments. Much of the early work on direct search methods arose in the statistics community. The connections between unconstrained optimization and statistical design of experiments are explored in [258]. For instance, Box's paper on evolutionary operation (EVOP) [35] proposed the construction of a two-level factorial design (aptly enough, a "box") around the current best known point in an effort to capture some local second-order effects. In $\mathbb{R}^2$, this gives the generating set

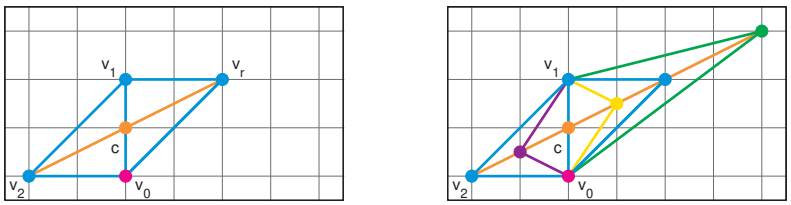
$$\mathcal{G} = \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix} \right\}. \tag{4.1}$$

Box also proposed augmenting this with multiples of the coordinate directions to obtain additional second-order effects. This was known as a composite design and can be represented easily enough within the context of GSS by augmenting the $\mathcal{G}$ in (4.1) with, for instance, the set

$$\mathcal{H} = \left\{ \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} -2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -2 \end{bmatrix} \right\}.$$

Both designs are illustrated in Figure 4.1. Further, recognizing that this approach can become onerous (since the number of points in a two-level factorial design is 2^n), fractional factorial designs were proposed as a way to capture some of the second-order effects without incurring the expense of quite so many function evaluations.

Seemingly in response to the high cost of two-level factorial designs, Spendley, Hext, and Himsworth [245] observed that a *simplex*, $n+1$ points in $\mathbb{R}^n$, is the minimal number of points required for a statistical design to capture first-order information. Note that this fits quite neatly with the fact that a generating set for $\mathbb{R}^n$ must contain a minimum of $n+1$ vectors. They explicitly proposed an optimization algorithm based on reflecting the vertex in the simplex with the highest function value (denote this vertex v_n) through the centroid of the opposite face ($c = \frac{1}{n}(v_0 + \cdots + v_{n-1})$). (See Figure 4.2(a) for an example in $\mathbb{R}^2$.) The simplex that results is accepted if f evaluated at the reflected vertex ($v_r = v_n + 2(c - v_n)$) is strictly less than f evaluated at the vertex in the simplex with the second highest function value (denote this vertex v_{n-1}). The rationale for this acceptance criterion is straightforward: if v_r becomes v_n at the next iteration (i.e., it has the highest function value in the new simplex), then the new reflection step will be the former worst vertex.



(a) (b)

Fig. 4.2 *Examples in* $\mathbb{R}^2$ *of both* (a) *the single possible step for the simplex algorithm of Spendley, Hext, and Himsworth and* (b) *the four basic steps for the Nelder–Mead simplex algorithm.*

The simplex algorithm of Nelder and Mead [194] is a variation on this basic idea that allows, in effect, a simple line search of the form $v_n + \alpha(c - v_n)$, with a set of four possible choices for α. Typical choices are $\alpha \in \{\frac{1}{2}, \frac{3}{2}, 2, 3\}$, as illustrated in Figure 4.2(b); see Lagarias et al. [164] for a particularly careful and complete description of the Nelder–Mead simplex algorithm. The line search has the effect of allowing the shape of the simplex to deform (for any choice of α *other* than 2), which is touted as a feature that allows the simplex to adapt to the local topology of the function, hence references to this algorithm as the *adaptive simplex method*.

While these two simplex algorithms are classical direct search methods, neither is a GSS method. These two simplex methods search along the *single* search direction $(c - v_n)$. Further, both these methods enforce only *simple* decrease, but in a sense that is subtly different from the definition of simple decrease defined in (2.3) and used in Step 2 of Algorithm 3.2, since their step acceptance condition requires simple decrease in f at the vertex in the simplex with the second highest function value v_{n-1}, *not* at the vertex in the simplex with the lowest known function value (denoted v_0). Thus both algorithms only ensure improvement of the function value at the sequence of *worst* vertices, but it is the sequence of *best* vertices (i.e., those with the lowest function value) that ultimately is of interest. As an interesting aside, the Nelder–Mead simplex algorithm *in* $\mathbb{R}^1$ can be restated as a GSS method. Since in this special case v_0 serves both as the centroid of the opposite face and as the vertex with the next highest function value, in effect the search is along two directions (a positive basis in $\mathbb{R}^1$) from v_0, and any improvement in the function value is with respect to f at v_0. An alternative proof for this special case, under different assumptions on f, is given in [164].

McKinnon [179] constructed a family of functions in $\mathbb{R}^2$ which demonstrates that the Nelder–Mead simplex algorithm can fail to converge to a stationary point of f, even if the family is parameterized so that f is strictly convex and has up to three continuous derivatives. Key to the failure McKinnon demonstrates is the ability to deform the simplex. Repeated deformations can cause the sequence of simplices produced by the Nelder–Mead simplex algorithm to converge to a *degenerate* simplex. Notice in Figure 4.2 that choosing $\alpha = \frac{1}{2}$ (a so-called *inside contraction*) replaces v_n with a vertex that moves closer to c. In McKinnon's examples, the Nelder–Mead simplex algorithm repeatedly chooses $\alpha = \frac{1}{2}$ (*only*) and the simplices converge to a straight line that is orthogonal to the steepest descent direction and have interior angles which tend to zero (i.e., the simplices collapse along the steepest descent di-

rection, a direction along which the goal should be for the simplices to enlarge). The collapse of the simplices means that the $n+1 = 3$ points in the limiting simplex do not span $\mathbb{R}^2$. Thus, the search is reduced increasingly to a one-dimensional hyperplane that does not contain the stationary point of the function.

McKinnon [179] includes a study of the effects of the choice of the starting simplex. He found that for some choices of the starting simplex, the behavior of the Nelder–Mead simplex algorithm applied to his family of functions, when parameterized to ensure that f is strictly convex, is not stable against small numerical perturbations caused by rounding error. He reports that in his numerical tests, rounding error was enough to prevent the algorithm from converging to the nonstationary point for the analytical examples he constructed. McKinnon hastens to add that this numerical behavior is very sensitive to the representation of the problem and to the details of the implementation of the Nelder–Mead algorithm (a sensitivity that others also have observed [129]) and of the function.

Nevertheless, there is long-standing anecdotal evidence that the Nelder–Mead simplex algorithm can fail in practice. There is additional experimental evidence [255] to suggest that the Nelder–Mead simplex algorithm can fail because the search direction becomes increasingly orthogonal to the direction of steepest descent. Recently, numerous modifications to the Nelder–Mead simplex algorithm have been posed in the mathematical optimization literature [153, 193, 224, 238, 262], but many users seem to believe that the easiest fix simply is to restart the search with a new simplex.

These old questions of what constitutes a good and effective experimental design have reappeared in the recent reemergence of model-based direct search methods, often referred to as *derivative-free optimization* (DFO) methods [71, 72, 74, 177, 220, 221]. DFO methods share many of the concerns of *response surface methodology* (RSM), originally proposed by Box and Wilson [38] for the purpose of optimizing a mean response function whose values can only be observed subject to random error. Because derivatives of the response function are not available, Box and Wilson sample the response function at designated points in a neighborhood of the current iterate and construct a local linear (or quadratic) model of the mean response via multiple linear regression. (A note on terminology: the RSM community calls the mean response function the *response surface*; the same phrase has been used by some practitioners of simulation-based optimization to describe a polynomial model of the objective function.) The primary difference between DFO and RSM is one of context: in RSM, fits are made using the method of least squares; in DFO, the function values are known and models are typically constructed via interpolation. DFO and RSM share a common concern with experimental design: the decision of where to sample the objective function is both crucial and unavoidable. The simplicity of working directly with the results of the sampling, rather than explicitly constructing an approximation, is what later led Box to propose EVOP [35] as an alternative approach to finding optimum conditions experimentally.

Much of the focus in the current research on sampling-based derivative-free methods addresses the question of the interaction between the quality of the model (particularly with respect to second-order information) and the so-called poisedness of the sample points used to construct the model. Powell [220] proposed using a simplex because of the fact that it contains the minimum number of points required to capture first-order effects. But added to this strategy is careful monitoring of the simplex to ensure that the simplex does *not* collapse into a lower-dimensional hyperplane. Alternatively, one can capture full second-order effects using $O(n^2)$ sample points, much as

was done in Glad and Goldstein [121], but start with a minimal linear model and build up to a full quadratic model as more and more sample points are accumulated during the course of the search [71]. Once again, monitoring the quality of the experimental design—the "poisedness" of the set of sample points—is critical. More recent efforts also have moved away from regression models to other forms of approximations, such as Newton interpolatory models [72] and support vector machines.

With the notable exception of Nelder–Mead, what all these strategies share is an emphasis on keeping the sample points "close" to the current iterate (recall Figure 4.1). The goal is to acquire the best possible local picture of f—first-order effects, at a minimum, and preferably full second-order effects—before making a decision about where to move next. The emphasis is on hedging all bets so that when a move is made away from the current iterate, it is made with a high degree of confidence that the trial step will succeed in producing decrease in f.

4.2. Exploratory Moves. Another philosophical approach to acceleration is to be less cautious and more speculative in the search. Specifically, aggressively sample the function at points that may be relatively far from the current iterate (where "relatively far" is with respect to the value of Δ_k).

Hooke and Jeeves called such speculative steps *exploratory moves* in connection with their pattern search algorithm [139]. To illustrate the exploratory moves, suppose that iteration $k-1$ was successful (no exploratory step is attempted when $k=0$). Then iteration k begins by conducting coordinate search about a trial point $x_p = x_k + (x_k - x_{k-1})$, rather than about the current iterate x_k. The idea is that since the step $x_k - x_{k-1}$ from x_{k-1} to x_k led to improvement in f, then further progress may be possible in the general direction of $x_k - x_{k-1}$. Such a step is called a *pattern step.* The objective is evaluated at the trial point x_p and the algorithm then proceeds to conduct coordinate search about x_p, even if $f(x_p) \geq f(x_k)$. If coordinate search about x_p is successful, and finds a point x_+ such that $f(x_+) < f(x_k)$, then x_+ is accepted as the new iterate x_{k+1}. If no such point is found around x_p, then the pattern step is deemed unsuccessful, and the method reduces to coordinate search about x_k.

In two dimensions, coordinate search about x_p involves first trying the point $x_p + \Delta_k e_1$. If $f(x_p + \Delta_k e_1) < f(x_k)$, then try $x_p + \Delta_k e_1 + \Delta_k e_2$; otherwise, try $x_p - \Delta_k e_1$. This continues until all n coordinate directions have been explored. The number of possible exploratory steps around x_p is $(3^n - 1)$, which, for our example in Figure 4.3, can be represented as

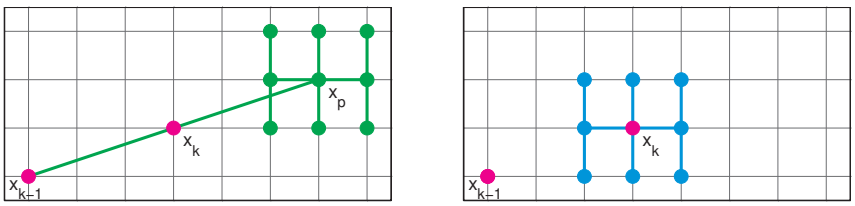
$$\mathcal{H}_k = \left\{ \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \end{bmatrix}, \begin{bmatrix} 4 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \begin{bmatrix} 3 \\ 0 \end{bmatrix} \right\}. \tag{4.2}$$

Note that *at least* n steps are tried (one along each coordinate direction), but *at most* $2n$ steps (two along each coordinate direction) are tried. Which steps are tried depends on whether the current trial step improved upon the best known value of f.

For the exploratory moves to succeed, at least one of either the trial point x_p or the trial points produced by the exploratory steps about x_p must produce decrease in f. If none of them does, the search falls back to an equivalent form of coordinate search about x_k, as shown in Figure 4.3(b). Thus, in $\mathbb{R}^2$,

$$\mathcal{G} = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right\}.$$

For the example in Figure 4.3, this requires augmenting $\mathcal{H}_k$ in (4.2) with the following



(a) (b)

Fig. 4.3 *Examples in* $\mathbb{R}^2$ *of both* (a) *possible exploratory moves for the pattern search algorithm of Hooke and Jeeves and* (b) *the fall-back strategy for the same algorithm.*

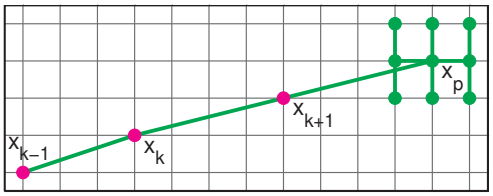


Fig. 4.4 *The gradual lengthening and reorientation of the steps and directions during a sequence of successful exploratory moves taken by the Hooke and Jeeves pattern search algorithm.*

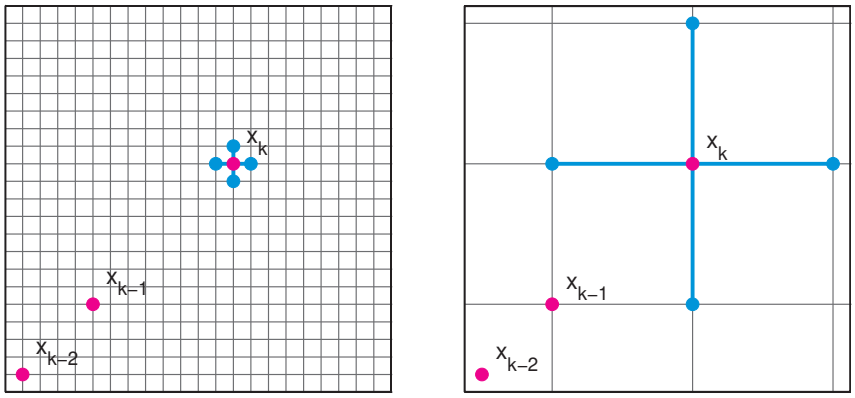
vectors:

$$\mathcal{H}'_k = \left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix} \right\}.$$

To appreciate the effect of the exploratory moves on the progress of the search, consider what could happen at the next iteration for the example in Figure 4.3 should one of the exploratory moves succeed, as illustrated in Figure 4.4. The cumulative effect of a sequence of exploratory moves is to lengthen the step taken at each iteration. Notice, also, the gradual modification of the direction taken. In the literature of the day such a strategy was referred to as "ridge following" (for maximization problems).

The important observation here—one that has informed some of the more recent algorithmic development for GSS methods—is that there are two classes of steps involved in the algorithm. The steps defined by Δ_k and the set $\mathcal{G}_k$ are the steps that are critical to ensure the overall convergence of the search process since the set $\mathcal{G}_k$ guarantees at least one direction of descent if x_k is not a stationary point of f. Hence the critical role played by $\mathcal{G}_k$ in the preceding analysis. But the directions in $\mathcal{G}_k$ need not be the ones used to define the next iterate. Instead the directions in $\mathcal{G}_k$ define a fall-back strategy; these are the directions to be tried (*polled* [91]) when more aggressive strategies fail. But this means that GSS algorithms are allowed to try any *finite* number of other steps, defined by Δ_k and $\mathcal{H}_k$, looking for a step that satisfies the decrease condition

$$f(x_k + \Delta_k d_k) < f(x_k) - \rho(\Delta_k). \tag{4.3}$$



(a) *when* $\Delta_k = \Delta_{k-1} = \Delta_{k-2}$ (b) *when* $\Delta_k > \Delta_{k-1} > \Delta_{k-2}$

Fig. 4.5 *Examples of a poll step in* $\mathbb{R}^2$.

If such a step can be found, then the *polling* can be deferred for at least one iteration since the iteration is a success (i.e., $k \in \mathcal{S}$). For algorithm design, the art then becomes finding an effective strategy for such exploratory moves.

This leads to the following observation. The lengths of the steps taken by the search may not necessarily be on the same scale as the length of the steps used to poll, particularly when an algorithm enforces the condition $\Delta_{k+1} \leq \Delta_k$, as is the case for the original pattern search algorithm of Hooke and Jeeves. Consider a recent string of successes produced by exploratory moves with steps that are quite long relative to Δ_k. However, once the exploratory moves do not succeed, the poll steps may be on a markedly different scale. Such a situation is illustrated in Figure 4.5(a).

The rational lattice strategy allows $\phi_k \geq 1$ when $k \in \mathcal{S}$. This means that after a successful iteration, it is possible to have $\Delta_{k+1} > \Delta_k$. A string of successes may be produced by exploratory moves with steps that are again long, but now on the same scale as Δ_k, and, once the exploratory moves do not succeed, the poll steps are on a scale that is related to the length of the most recent successful step. Such a situation is illustrated in Figure 4.5(b).

As a consequence of the fact that ϕ_k may be greater than 1 when $k \in \mathcal{S}$, for the rational lattice strategy it is only possible to show that $\liminf_{k\to\infty} \Delta_k = 0$, not $\lim_{k\to\infty} \Delta_k = 0$, holds (see [10] for an example illustrating why this is so).

This discussion also helps illustrate one other feature of GSS methods and their accompanying analysis, and that is the flexibility available for defining the *length* of steps. There actually are *two* components that define the length of the step: Δ_k and $\|d\|$, where $d \in \mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$. For the analysis—and even much of the discussion here—it is convenient to focus on what happens as Δ_k is varied; in (3.4), imposing uniform bounds on the length of $\|d_k\|$ makes this focus possible. But other options for monitoring $\|\Delta_k d_k\|$ are discussed further in section 5.1.

4.3. Using Oracles. Another recent development in terms of algorithmic strategies further exploits the distinction between the steps defined by Δ_k and $\mathcal{G}_k$ and those defined by Δ_k and $\mathcal{H}_k$. Suppose that for all k one were willing to choose

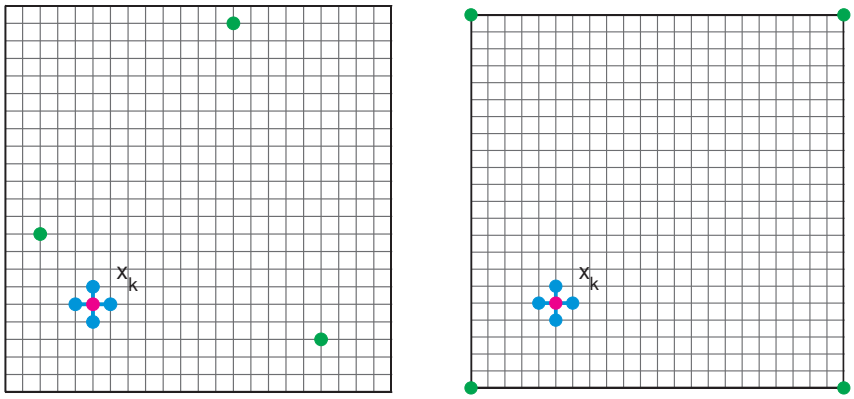


Fig. 4.6 *Two examples in* $\mathbb{R}^2$ *of using oracles to try to predict successful steps.*

$\mathcal{G}_k = \{\pm e_i \mid i = 1, \ldots, n\}$. Now suppose that $\mathcal{H}_k$ were "everything else" for some bounded portion of the mesh near x_k. Figure 4.6 demonstrates this in $\mathbb{R}^2$ with

$$\mathcal{G}_k = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right\}.$$

and

$$\mathcal{H}_k = \left\{ \begin{bmatrix} -5 \\ -5 \end{bmatrix}, \begin{bmatrix} -4 \\ -5 \end{bmatrix}, \begin{bmatrix} -3 \\ -5 \end{bmatrix}, \ldots, \begin{bmatrix} 15 \\ 17 \end{bmatrix}, \begin{bmatrix} 16 \\ 17 \end{bmatrix}, \begin{bmatrix} 17 \\ 17 \end{bmatrix} \right\}.$$

As usual, x_k is shown in magenta and the poll steps defined by Δ_k and $\mathcal{G}_k$ are shown in blue. The job of the *oracle* is to choose some finite subset of the steps defined by Δ_k and $\mathcal{H}_k$ to try as part of the exploratory phase of the search. In Figure 4.6, two possible sets of choices are indicated in green. Once again, the search only falls back on the poll steps if the initial explorations are unable to produce a step that satisfies the decrease condition (4.3). From an analytical standpoint, it simply does not matter how the oracle makes her choices, so long as the process is finite and the usual fall-back strategy is in place. So the oracle can "guess"; she can resort to some elaborate—or even not so elaborate—algorithmic strategy; or she can rely on divine providence—hence the name. In truth, the oracle is nothing more or less than a predictor; her job is to predict a point (or a finite set of points) at which one might observe decrease in the value of f at x_k.

The distinction between the exploration phase and the polling phase was noted in [91]. There it was observed that if, for instance, the underlying function f were a partial differential equation whose value was computed using some finite discretization, perhaps a coarser—and thus less expensive to compute—discretization could be used to predict points at which the finer discretization would produce decrease.

Homage for this viewpoint is due, in part, to Hooke and Jeeves [139] since the necessary analytical tools in [257] were constructed to handle the exploratory moves in their algorithm. In this context, the oracle for Hooke and Jeeves is their pattern step, which assumes, when $k - 1 \in \mathcal{S}$, a simple one-dimensional linear model of f involving x_{k-1} and x_k. The coordinate steps taken about the trial point x_p could be

viewed as an acknowledgment that the implicit one-dimensional linear model is not likely to be particularly accurate, so further modest adjustments may be required.

This leads to the second observation, which is of further practical import. In both instances, a *surrogate* of the objective is used to predict success (i.e., the coarser discretization of the PDE or the simple one-dimensional linear model). And in both instances, the oracle attempts to exploit additional knowledge of the objective function f—knowledge other than the gradient—in an attempt to accelerate the search. In the first instance, it is in the form of an appeal to the numerical approximation underlying the problem, but in a less costly computational form. In the second instance, it is in the form of an attempt to build some sense of "history" into the search by using function information obtained at earlier iterations to inform predictions about where to look next.

The latter viewpoint, in particular, has been pursued by using algebraic approximations of f built using previously computed values of $f(x)$ [30, 31, 239, 261]. The assumption is that the evaluation of $f(x)$ is computationally expensive, so the idea is to put previous evaluations to work. An algebraic approximation is constructed to be inexpensive to compute so that it can be used as a predictor. The oracle then is a process that (inexactly) optimizes the algebraic approximation of f and returns the (inexact) optimizer of the algebraic approximation as a point at which improvement in f may be observed.

In some sense this is a return to the "inner iteration–outer iteration" perspective for line search algorithms outlined in section 2.2. At the outer iteration of a line search algorithm, an inexpensive-to-compute local polynomial model of the function is built using the gradient of f at x_k. The inner iteration of a line search algorithm involves finding (inexactly) the minimizer of the local polynomial model. The (inexact) minimizer of the model is then returned as a prediction about where decrease in f may be found. In the realization that the local polynomial model is not usually likely to be particularly accurate, a globalization strategy—in the form of a backtracking line search—may need to be employed to actually produce decrease.

For this surrogate-based approach to GSS methods, at the outer iteration, an inexpensive-to-compute algebraic approximation of the function is built using previously computed values of f. The inner iteration involves finding (inexactly) at least one minimizer of the algebraic approximation. A finite number of (inexact) minimizers of the algebraic approximation then are returned as predictions about where decrease in f may be found. In the realization that the algebraic approximation is not usually likely to be particularly accurate, a globalization strategy—in the form of the poll steps—may need to be employed to actually produce decrease.

Observe that the notion of an oracle also applies to the line search methods discussed in section 2.2. Any point satisfying the Armijo–Goldstein–Wolfe conditions (2.4)–(2.5) is acceptable as a next iterate. In practice, the next step is generated by a line search along a descent direction since this procedure is guaranteed to produce an acceptable next iterate. However, one is perfectly free to try any points one wishes, so long as no point is accepted without checking that it satisfies (2.4)–(2.5).

5. Variations on the GSS Requirements. The goal of the GSS framework given in section 3.3 was to give a specification that was generic enough to capture many algorithmic variations in the literature while at the same time being simple enough to derive convergence results with a minimum of notation and special cases. In doing so, we necessarily excluded some interesting and useful variations that recently have appeared in the literature. In this section we expand, briefly, on some of these variations and point the interested reader to the appropriate references for further study.

5.1. Controlling the Step Length. GSS prescribes a particular regimen for updating the step-length control parameter—it must be reduced after an unsuccessful iteration and can be increased only after a successful iteration. Along with other suitable safeguards, this way of updating Δ_k ensures that the methods can be *globalized*; i.e., that the following holds:

$$\lim_{\substack{k \to +\infty \\ k \in \mathcal{K} \subseteq \mathcal{U}}} \Delta_k = 0. \tag{5.1}$$

The regimen for updating the step-length control parameter within the GSS framework can be relaxed, so long as (5.1) holds. For example, the following modification does not fit within the GSS framework, but it does lead to equivalent convergence results. Let $\phi_k = 1$ for all $k \in \mathcal{S}$ (no expansion steps), and let θ_k for $k \in \mathcal{U}$ be defined *in any way* (even $\theta_k \geq 1$ is admissible) that ensures that (5.1) holds. Variations on this idea can be incorporated into the three globalization strategies discussed in section 3.7 to produce equivalent first-order convergence results. The work of Coope and Price [77] motivated the discussion of moving grids in section 3.7.3, but the specifications on the algorithms given in their paper are more general and encompass this alternate criteria for the update of the step-length control parameter.

Another variation, common to the sufficient decrease approaches in [115, 174], is the following. Given a fixed set of directions $\mathbf{G} = \{d^{(1)}, \ldots, d^{(p)}\}$ (as in (3.18)), associate a separate step-length control parameter $\Delta_k^{(i)}$ with each search direction and then update the p step-length control parameters independently. This allows the directions to be scaled individually, each adapting to the curvature of the underlying objective function.

The rational lattice approach also is amenable to the use of semiautonomous step-length control parameters, as outlined in [161, 162]. In this setting, the semiautonomous behavior of the $\Delta_k^{(i)}$ is introduced to address computational load-balancing issues, rather than to capture curvature information.

This idea of using separate step-length control parameters brings us to the more general question of controlling the length of the step. As mentioned at the close of section 4.2, the steps at iteration k in GSS are composed of two components: the step-length control parameter Δ_k and the search directions $\|d\|$, $d \in \mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$. We bound the lengths of the search directions as

$$\begin{array}{ll} \beta_{\min} \leq \|d\| \leq \beta_{\max} & \text{for all } d \in \mathcal{G}_k, \text{ and} \\ \beta_{\min} \leq \|d\| & \text{for all } d \in \mathcal{H}_k, \end{array} \quad \text{for } k = 1, 2, \ldots. \tag{5.2}$$

By enforcing (5.2), we have that the length of any step $\|\Delta_k d\|$ for $d \in \mathcal{G}_k$ is bounded as a constant multiple of Δ_k,

$$\Delta_k \beta_{\min} \leq \|\Delta_k d\| \leq \Delta_k \beta_{\max}.$$

Likewise, the lower bound holds for $d \in \mathcal{H}_k$. By bounding the lengths of the search directions as in (5.2), parts of the algorithm and accompanying theory that depend on the lengths of the steps can be written in terms of the step-length control parameter Δ_k alone.

Yet another variation would be to eliminate the step-length control parameter Δ_k entirely and work directly with the lengths of each vector in $\mathcal{D}_k$. Doing so,

however, would necessitate a substantially different development of the notation and presentation. For example, (5.1) would be replaced with something like

$$\lim_{\substack{k\to+\infty \\ k\in\mathcal{K}\subseteq\mathcal{U}}} \max_{d\in\mathcal{G}_k} \|d\| = 0. \tag{5.3}$$

5.2. Additional Algorithmic Steps. As discussed in section 4, many methods fit the GSS framework by choosing the set $\mathcal{H}_k$ in a particular fashion. Since the choice of $\mathcal{H}_k$ affects only the successful steps, it is largely tangential to the theory.

Depending on the particular method of globalization, it may be possible to add special additional steps that give the algorithm another chance to find an improving point, perhaps subject to different rules.

For example, the grid-based optimization framework of Coope and Price [77] motivated the moving grids approach described in section 3.7.3, but their approach adds something extra. It allows for a "finite process" after each *unsuccessful* iteration. In terms of GSS, one might think of this as follows. After each unsuccessful step, allow a special iteration where only $\mathcal{H}_k$ need be defined and has no restrictions and Δ_k is *not* modified even in the event of an unsuccessful iteration. In this case, Theorem 3.9 is the same except that $\omega(k)$ is replaced everywhere by $\omega(k)+1$. Though this does not fit the GSS framework, the basic ideas are the same.

Likewise, Lucidi and Sciandrone [174] have a similar extra step in a sufficient decrease algorithm that is close to the description in section 3.7.1. Their method allows a special iteration after *every* step, successful or unsuccessful. As before, only $\mathcal{H}_k$ need be defined and Δ_k is not modified. Furthermore, this special step requires only *simple* decrease to accept the iterate.

In both cases, these special steps only allow additional successes and are implemented in ways that respect globalization. Practically speaking, these special steps are another way to incorporate a surrogate without affecting the convergence properties.

5.3. Independent Directions for Parallel Computing. The asynchronous parallel pattern search (APPS) method [140] forms a generating set and farms out each direction to an independent process in a parallel or distributed computing environment. The unique feature of APPS is that it dynamically makes decisions without waiting for the information on all processors to be available. This can eliminate substantial idle time in the computation. Convergence is detected when the subset of processes that are done (all sharing a common accumulation point) forms a generating set. Since the algorithm is asynchronous, this means that convergence may be detected, and the search terminated, even though the remaining processes have not necessarily finished the search along their designated directions. Convergence of this method is proved by exploiting the properties of GSS in a slightly different context [161].

6. The Role of Smoothness in the Robustness of GSS Algorithms. The assumption that f is continuously differentiable is critical in showing that a subsequence of unsuccessful iterates converges to a stationary point of f; see Theorem 3.11. On the other hand, there is the conventional wisdom, nicely expressed by Swann [250] in his 1972 survey:

> The primary reason for interest in direct search methods is their generality; they require only the ability to evaluate the objective function at specified points of $[\mathbb{R}^n]$, and since they usually make no assumptions other than continuity of the function they can be applied to a very wide class of

> problems. They are thus particularly useful for cases where the function is non-differentiable, where the first partial derivatives of the function are discontinuous, and where the function value depends upon some physical measurement and may therefore be subject to random error, all of which are problems which can cause difficulties using the more theoretically based gradient methods.

This view is echoed in Gill, Murray, and Wright [117]:

> Methods based on function comparison are intended only for problems in which $[f(x)]$ is discontinuous, the gradient vector $g(x)$ is discontinuous at the solution, or $g(x)$ has so many discontinuities that a method assuming continuity of $g(x)$ will fail. Furthermore, the discontinuities are presumed to have no special structure.

In fact, certain qualities of GSS methods do lend themselves to problems such as those described above, as we discussed in section 1.2. In this section, we discuss why it is reasonable to expect GSS methods (and other direct search methods) to work in practice on badly behaved problems such as that depicted in Figure 1.2. We go on to point out, however, that there are limits to what one can prove and hope for in practice when applying direct search methods to nonsmooth problems.

6.1. The Effects of Multiple Search Directions and Sampling. A strength of GSS methods, as compared to many other direct search methods, is the fact that GSS methods look in multiple directions at every iteration. This means that even if f is nondifferentiable at an iterate x_k, so that there is no guarantee that the set of search directions contains a descent direction, the chance of finding a point with a lower objective value is greater for GSS than for methods that only consider a single direction.

Another advantage that comes about by using multiple search directions is that GSS methods are not as easily misled by "noise" in the objective function. Consider, for example, the objective function pictured in Figure 1.2. Here, GSS methods will easily realize decrease in the function so long as the step size is large relative to the noise. Derivative-based methods, on the other hand, may have a hard time making any progress at all because the local descent directions may actually point uphill when we consider the overall trend of the function. In other words, while one of the GSS search directions is guaranteed to be a descent direction, we may actually take the step produced by a direction of local *ascent* because with the right choice of Δ it produces decrease in f. In a line search method one takes care that the single search direction is a descent direction, since, in the event of backtracking, we must be guaranteed decrease in f along this direction.

Another aspect of GSS methods that helps explain their robust behavior is the smoothing effect of the process by which trial steps are generated. This is particularly the case when GSS methods are applied to functions such as the objective in Figure 1.2, which is afflicted with low-amplitude, high-frequency "noise" that would be amplified by differentiation. The smoothing effect described here also plays a role in methods such as those in [154, 178, 181] that use purposefully coarse finite-difference approximations to derivatives as a means of treating noisy objectives.

GSS methods perform a type of sampling. A familiar consequence of sampling is that high-frequency features on a length-scale finer than the sampling are not seen, but instead appear in an aliased form as part of a smoother version of the function. The smoothing effect of the sampling done in GSS is explained most clearly for compass search, because the trial points lie on uniform Cartesian grids.

Without loss of generality, assume that $x_0 = 0$. Then, if Δ_k is the current step size parameter, any potential iterate is of the form $\Delta_k v$, where $v \in \mathbb{Z}^n$ (see

Theorem 3.7). Let $\hat{f}$ be the discrete Fourier transform of f on the lattice $\Delta_k\mathbb{Z}^n$: for $\omega \in [-\pi/\Delta_k,\ \pi/\Delta_k]$ [247],

$$\hat{f}(\omega) = \frac{1}{(2\pi)^{n/2}} \sum_{v\in\mathbb{Z}^n} e^{-i\Delta_k v\cdot\omega}\ f(\Delta_k v)\ \Delta_k^n.$$

Consider the band-limited interpolant f_{Δ_k} of f over $\Delta_k\mathbb{Z}^n$ defined by

$$f_k(x) = \frac{1}{(2\pi)^{n/2}} \int_{[-\pi/\Delta_k,\ \pi/\Delta_k]^n} e^{ix\cdot\omega}\ \hat{f}(\omega)\ d\omega.$$

Then

$$f(\Delta_k v) = f_k(\Delta_k v) \qquad \text{for all } v \in \mathbb{Z}^n.$$

Because the algorithm only uses the values of f on the grid $\Delta_k\mathbb{Z}^n$, its behavior will be the same as if it were applied to the band-limited (and thus smoother) interpolant f_k. For larger values of Δ_k, this interpolant lacks the small-amplitude, high-frequency oscillations present in Figure 1.2. The sampling of f is too coarse to "see" oscillations that occur on a fine length-scale. Features of f corresponding to frequencies of magnitude larger than π/Δ_k are not captured, so the GSS algorithm is untroubled by them.

This leads to an interpretation of GSS algorithms as continuation methods in Δ. GSS starts with a smooth approximation f_0 of f and finds an approximate minimizer of f_0. Because GSS looks in multiple directions, one of which is a descent direction for f_k, we can expect to make progress on minimizing f_k. Moreover, the fact that f_0 may be considerably smoother than f should make this an easier problem to solve. As Δ_k tends to 0, higher frequency features of f are resolved. One may reasonably expect that the local minimizers with larger basins of attraction are found for larger values of Δ_k, thus avoiding the numerous uninteresting local minimizers such as those present in Figure 1.2.

The multiple search directions and the sampling may be viewed as complementary in the following way. Suppose that in compass search we actually evaluate f at all $2n$ points of the form $x_k \pm \Delta_k e_i$. If the sampling is coarse enough, i.e., Δ_k is larger than the length-scale of the "noise," then GSS escapes the effect of the "noise." Rather than form a centered finite-difference gradient, thereby reducing the information from these multiple directions to a single direction, GSS uses the fact that at least one of the search directions is guaranteed to be a descent direction. GSS thus avoids any problem with truncation error in finite-differencing, which would be present since Δ_k is presumed to be not too small.

The use of multiple search directions and longer trial steps explains why GSS methods have proven robust and effective in practice, and thus, in part, why they have remained popular with users. These features of GSS methods are of particular help for objectives like the one in Figure 1.2, where there is low-amplitude, high-frequency "noise" with no discernible structure. In this regard we would second the recommendation of Gill, Murray, and Wright in the passage quoted above in prescribing direct search methods for problems in which "the discontinuities are presumed to have no special structure."

That said, users of GSS methods should be aware of the potential pitfalls. In particular, GSS methods may not converge to a stationary point when applied to nonsmooth problems—especially when the loci of nonsmoothness are highly structured.

The following analysis and examples make clear that even though GSS methods *can* be used (i.e., they are well-defined) on nonsmooth problems, there is no theoretical guarantee that they will succeed.

6.2. Why We Assume f Is Continuous. One often encounters discontinuous functions in simulation-based design. Sometimes the discontinuities are a result of features (such as table look-ups and switches) in simulation codes. Other times, they are due to unexpected failures of the simulation codes. For these reasons, it is worth considering what can happen when GSS methods are applied to discontinuous problems. Frequently they enjoy success because the sequence of iterates "steps over" the loci of discontinuity.

However, although GSS *may* still find a stationary point if f is not continuous, there is no *guarantee* that it will do so. Consider the following one-variable example, where f is defined to be

$$f(x) = \begin{cases} (x-1)^2 & \text{if } x \neq 0, \\ 5 & \text{if } x = 0. \end{cases}$$

Suppose compass search begins at $x_0 = -1$ with $\Delta_0 = 1$. Furthermore, suppose we choose $\mathcal{D}_k = \{-1, +1\}$, $\phi_k = 1$, and $\theta_k = \frac{1}{2}$ for all k. Then every odd iteration is a contraction and even iterations are of the form

$$x_{2k} = 2^{-k}.$$

This is illustrated in Figure 6.1. Here $x_k \to 0$, which is *not* a stationary point of f.

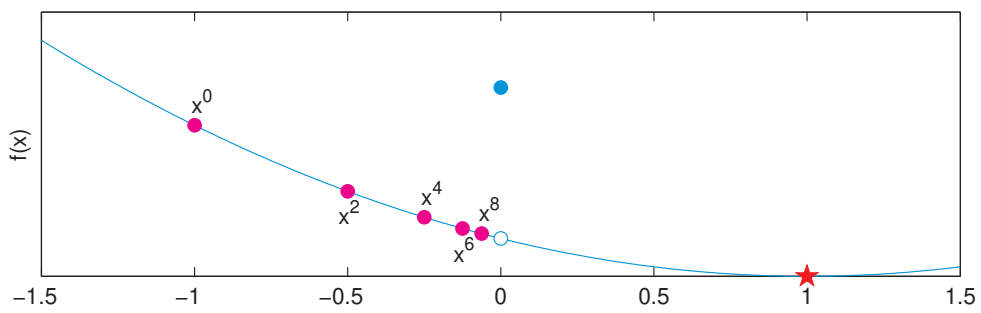
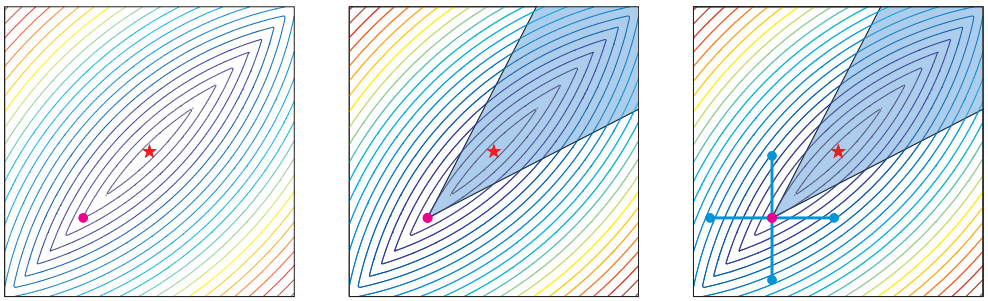


Fig. 6.1 *Compass search converging to a nonstationary point of a discontinuous function.*

Certainly this is a contrived example. Choosing $x_0 = -1.1$ would avoid these problems altogether because compass search would never even evaluate the function at $x = 0$. Although an identical example could easily be produced for a gradient-based algorithm, we stress this point because direct search methods are so often prescribed for discontinuous functions.

6.3. Why We Assume ∇f Exists and Is Continuous. When $f(x)$ is continuously differentiable, GSS methods work because, at every iteration, one of the search directions is a descent direction. That is, GSS methods are guaranteed to search along at least one direction within 90° of $-\nabla f(x)$.

There is no guarantee that a descent direction will exist in the nonsmooth case, as the next example shows.



(a) (b) (c)

Fig. 6.2 *A variant of the Dennis–Woods function.*

Consider the following variant of the Dennis–Woods function [94]:

$$f(x) = \frac{1}{2} \max \left\{ \|x - c_1\|^2, \|x - c_2\|^2 \right\}, \tag{6.1}$$

where $c_1 = (1, -1)^T$ and $c_2 = -c_1$. The level sets of this function are shown in Figure 6.2(a), with the minimizer indicated by a red star. This function is continuous and strictly convex, but its gradient is discontinuous everywhere on the line $x_1 = x_2$. In [94], it was shown that a modification of the Nelder–Mead simplex algorithm can fail to converge to the minimizer of f.

Now consider a point of the form $(a, a)^T$, with $a \neq 0$. An example is shown as a magenta dot in Figure 6.2. The set of directions from $(a, a)^T$ along which f initially decreases generates the shaded region. The objective increases along any direction outside this region. Figure 6.2(c) depicts the search directions for compass search emanating from $(a, a)^T$. Note that although $(a, a)^T$ is not stationary, none of the directions in the generating set can yield decrease for any value of the step-length control parameter Δ_k. In fact, numerical tests show that compass search applied to (6.1) frequently converges to a point of the form $(a, a)^T$ with $a \neq 0$.

In [256], this sort of failure was observed for the multidirectional search (MDS) algorithm, a type of GSS method. There it was shown that, under the assumptions that $\mathcal{L}_f(x_0)$ is compact and that f is continuous on $\mathcal{L}_f(x_0)$, some subsequence of the sequence of iterates produced by MDS converges to a point x_* in a set $\mathcal{X}_*$. The set $\mathcal{X}_*$ is the set that contains all the stationary points of f on $\mathcal{L}_f(x_0)$, all points in $\mathcal{L}_f(x_0)$ where f is nondifferentiable, and all points in $\mathcal{L}_f(x_0)$ where the derivative of f exists but is not continuous.

6.4. GSS Methods and Lipschitz Optimization. Now let us consider the general question of applying GSS to the minimization of functions that are Lipschitz continuous but fail to be differentiable in the usual sense at some points. For Lipschitz functions there is the notion of a generalized gradient and an attendant differential calculus [66]. A particular motivation for studying Lipschitz functions here is that some of the GSS approaches to nonlinear constraints (e.g., penalization) lead to unconstrained, Lipschitz minimization problems (see section 8.3.3).

For a function f that is Lipschitz near a point x, the (Clarke) generalized directional derivative in the direction v, denoted by $f^\circ(x;v)$, is defined to be

$$f^\circ(x;v) = \limsup_{\substack{y \to x \\ t \downarrow 0}} \frac{f(y+tv) - f(y)}{t}.$$

For many nonsmooth (convex or max) functions, including the example given in (6.1), the generalized directional derivative coincides with the usual directional derivative. The (Clarke) generalized gradient $\partial f(x)$ is

$$\partial f(x) = \left\{ \zeta \mid f^\circ(x;v) \geq \zeta^T v \text{ for all } v \right\}.$$

If f is decreasing from x in the direction d, then $f^\circ(x;d) \leq 0$. This means that for minimization, one is interested in finding directions d for which the generalized directional derivative is negative. As observed in section 3.4.1 for the smooth unconstrained case, at least one GSS search direction must be a descent direction because any half-space contains at least one direction in the generating set. However, a fundamental difficulty of nonsmooth optimization is that the set of descent directions can be smaller than a half space. Thus, to be assured of finding a successful step, a GSS method would need to have a search direction in the interior of this set. It is not clear how to ensure this without knowledge of $\partial f(x)$.

The assumption that f is continuously differentiable is used in the convergence analysis of GSS in two ways. First, it allows the use of the mean value theorem, as in the proof of Theorem 3.3. Second, it makes it possible to take the limit of the gradient as $k \to \infty$, as in Theorem 3.11. If f were only Lipschitz, the generalized mean value theorem could be applied, but the continuity needed to take the limit of the gradient as $k \to \infty$ would be absent.

In the case of Lipschitz functions, the negative result from [256] mentioned at the end of the previous section is refined in [13]. There it is shown that under some mild assumptions, if the sequence of points generated by a GSS algorithm converges to a point x_* and f is Lipschitz near x_*, then for all search directions d

$$f^\circ(x_*, d) \geq 0. \tag{6.2}$$

This describes the situation in the example of the previous section: at the point $x_* = (a, a)^T$ in Figure 6.2, the function is increasing along each of the search directions.

However, unless $\partial f(x_*)$ is a singleton, (6.2) does not necessarily mean that x_* is a stationary point (i.e., $0 \in \partial f(x_*)$). If one assumes in addition that f is strictly differentiable at x_*, then $\partial f(x_*)$ reduces to a singleton and x_* is a stationary point, as noted in [13]. A function f is strictly differentiable at x if there exists a vector w such that for all v,

$$\limsup_{\substack{y \to x \\ t \downarrow 0}} \frac{f(y+tv) - f(y)}{t} = w^T v.$$

The following example illustrates this latter result. We modify (6.1) to make it strictly differentiable at its minimizer. Define $\hat{f}$ as

$$\hat{f}(x) = \left(1 - e^{-\alpha \| x \|^2}\right) f(x),$$

where $\alpha = 10^3$ and $f(x)$ is as in (6.1). A picture of $\hat{f}$ is given in Figure 6.3.

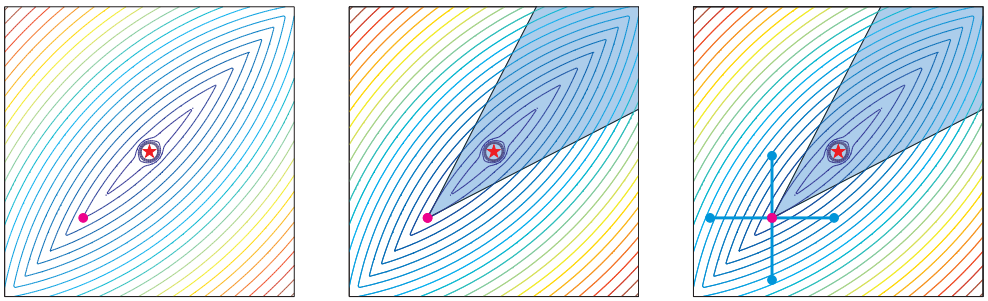


Fig. 6.3 *A slightly smoothed variant of the Dennis–Woods function.*

Once again, the minimizer is at $x_* = (0,0)^T$ but now, by Proposition 2.3.13 in [66], $\hat{f}$ is strictly differentiable at x_*. The strict differentiability result from [13] says that if a GSS method were to converge to $(0,0)^T$, then, because $\hat{f}$ is strictly differentiable at $(0,0)^T$, $(0,0)^T$ must be a stationary point. GSS has no guarantee, however, of converging to $(0,0)^T$; in fact, just as for (6.1), compass search tends to converge to a point of the form $(a,a)^T$, with $a \neq 0$.

On the other hand, functions such as $f(x) = \| x \|$ are not strictly differentiable at their minimizers, nor are the exact penalty functions in section 8.3.3 where inequality constraints are binding. Thus, alas, there remains a significant gap in theory and practice between what we would like to be true about GSS methods and what is actually true.

6.5. The Gap between Theory and Practice. GSS methods (or any direct search method) can be applied in a straightforward way to just about any optimization problem, smooth or nonsmooth. The fact that GSS methods search in multiple directions helps reduce the sensitivity of these algorithms to nonsmoothness. Moreover, the sampling done in GSS can frequently step over loci of nonsmoothness, especially if the nonsmoothness has no particular structure.

We also note that many users are satisfied even when a stationary point cannot be guaranteed. Finding a stationary point of an objective like that in Figure 1.2 might not be what is really wanted. Instead, appreciable improvement in the objective function, at a reasonable cost of time and human effort, is sufficient for many applications.

This said, an inescapable conclusion of this section is that currently there are no theoretical guarantees that GSS methods will succeed on nonsmooth problems. In both the examples set forth above and in our practical experience, there are situations where GSS methods do, in fact, fail to find a stationary point.

Nevertheless, GSS methods, like many optimization algorithms, perform better in practice than their supporting theory might suggest.

7. Some Background on Constrained Optimization. The constrained optimization problem of interest is

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && c(x) \leq 0. \end{aligned} \tag{7.1}$$

Here $c : \mathbb{R}^n \to \mathbb{R}^m$, and f and c are assumed to be continuously differentiable. The inequality $c(x) \leq 0$ is interpreted to hold term by term; i.e., $c^{(i)}(x) \leq 0$, $i = 1, \ldots, m$.

The set of points satisfying the constraints in (7.1) is called the *feasible set* and is denoted by Ω:

$$\Omega = \{ x \mid c(x) \leq 0 \}.$$

A point x is *feasible* if $x \in \Omega$.

A *feasible iterates* algorithm ensures (and may require) that each iterate produced is feasible. An *infeasible iterates* approach, on the other hand, allows $x_k \notin \Omega$ for some iterations, and only requires feasibility in the limit. In the case of simulation-based optimization, it sometimes is necessary to use methods that produce only feasible iterates since the simulations involved may make no provision for computing outputs if the inputs x are outside the feasible region.

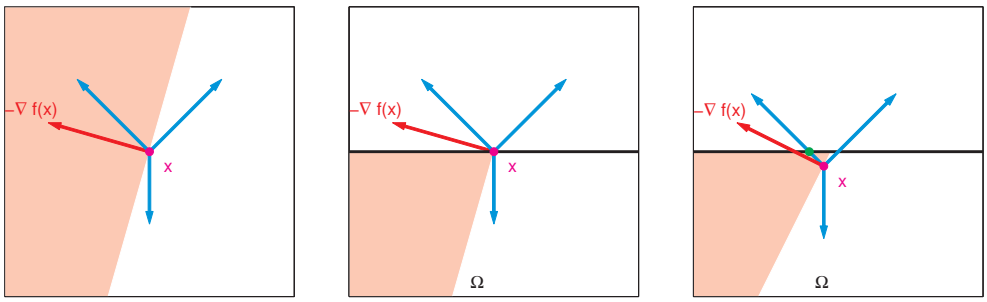
Historically, direct search methods took a feasible iterates approach. Most of the early development of direct search methods for problems with constraints predated the development of infeasible iterates methods such as augmented Lagrangian algorithms [135, 215]. But recall, too, that a common situation in practice is that one wants "improvement" rather than full-blown optimality. Since the user's goal may be only one or two correct digits, satisfying feasibility "in the limit" is not acceptable. Furthermore, in the case of real physical experiments, violating certain constraints (say, setting a temperature to less than absolute zero) may lead to nonphysical, meaningless results (see [35] for a discussion of one such example in the context of chemical plant management).

Our discussion focuses primarily on methods that use a feasible iterates approach; methods that adopt an infeasible iterates approach are presented, briefly, in section 8.3.

7.1. The Effects of Constraints. In the unconstrained case, it suffices for one of the search directions to lie within 90° of the direction of steepest descent, as discussed in section 2.2. However, if there are constraints present, the situation is more complicated because the constraints may prevent the search from taking a feasible step that would decrease the objective function. Obtaining a direction that yields improvement in the objective while not violating the constraints limits the choices of suitable search directions.

This is illustrated in Figure 7.1(a)–(b). In (a), the red arrow indicates the direction of steepest descent from the point x indicated by the magenta dot. The blue directions form a positive basis and thus would be an acceptable set of search directions (a generating set) for a GSS method in the unconstrained case. In (b) a single constraint, which restricts the solution to the lower half of the figure, is introduced. The boundary of the feasible region Ω is indicated with a black line. To obtain a direction that is within 90° of the direction of steepest descent *and* which does not carry the search out of the feasible region, the directions are constrained to those that fall in the shaded region. Unfortunately, none of the search directions are acceptable in this case—either they are not directions of descent or they take the search outside of the feasible region.

Equally annoying is the situation in Figure 7.1(c). Now x is in the interior of the feasible region Ω, but close to the boundary. The direction pointing toward 10 o'clock is a descent direction, but the search can only take a very short step along this direction if the step is to remain feasible. The longest step possible is indicated by the dot shown in green. As a consequence, only a small improvement in the objective may be realized, much smaller than is possible while remaining feasible. This is solely because of the interaction between the choice of search directions and the geometry



(a) A descent direction in the generating set for the unconstrained case.

(b) With the introduction of a constraint, no *feasible* descent directions in this set.

(c) Feasible step along the one descent direction is too short.

Fig. 7.1 *How constraints can limit acceptable search directions.*

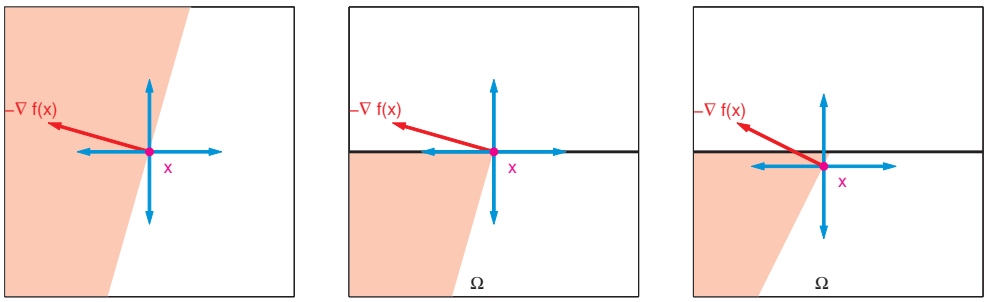
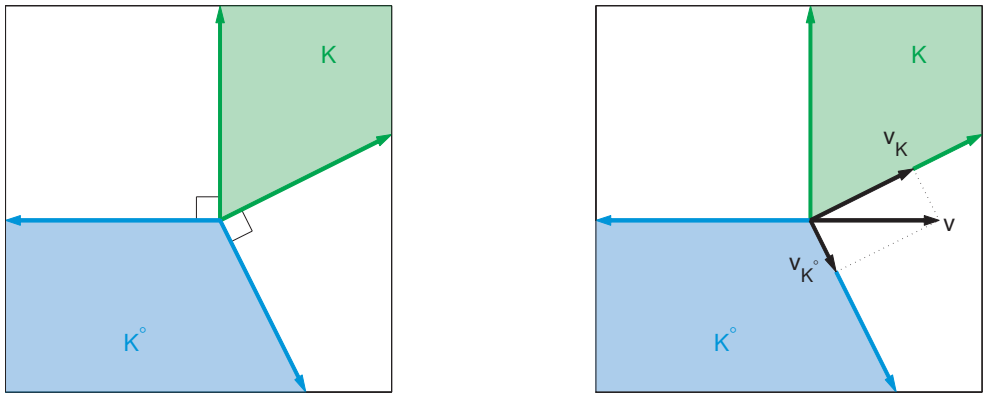


Fig. 7.2 *At least one (feasible) direction of descent along which a sufficiently long step can be taken.*

of the nearby portion of the boundary. These short steps, in turn, could cause an unwary algorithm to converge to a point on the boundary that is not a constrained local minimizer. The situation depicted in (c) is similar to the problem of bad step lengths discussed in connection with line search methods in section 2.2; i.e., the steps are short even though the search may be far from any minimizer.

A moment's reflection reveals that the difficulty derives from the fact that the generating set does not include steps parallel to the boundary. What is needed are directions that allow the search to move along any nearby part of the boundary, taking feasible steps that are sufficiently long. In fact, the remedy for the constraint illustrated in Figure 7.1 is surprisingly simple, as can be seen in Figure 7.2.

7.2. The Geometry of Cones. Unconstrained and equality constrained optimization can be described in terms of the geometry of linear subspaces. So for the unconstrained case, a generating set for $\mathbb{R}^n$ is the same as a positive spanning set, making it possible to write any vector in $\mathbb{R}^n$ as a linear combination of the vectors in the generating set, with the additional requirement that all the coefficients be nonnegative. This was the import of Definition 3.1 in section 3.4.



(a) A cone K and its polar K°.

(b) The projection of v onto K and its polar.

Fig. 7.3 *An example of the geometry of cones in* $\mathbb{R}^2$.

Inequality constrained optimization, as in (7.1), requires the more complicated geometry of cones. A set K is a *cone* if for any $x \in K$, $\alpha x \in K$ for all scalars $\alpha \geq 0$. Definition 3.1 is the specialization of the the following definition to the case when $K = \mathbb{R}^n$ [86].

DEFINITION 7.1. *Let $\mathcal{G}$ be a set of p vectors in $\mathbb{R}^n$: $\mathcal{G} = \{d^{(1)}, \ldots, d^{(p)}\}$. Then the set $\mathcal{G}$* generates *the cone K if*

$$K = \left\{ v \mid v = \sum_{i=1}^{p} \lambda^{(i)} d^{(i)},\ \lambda^{(i)} \geq 0 \text{ for } i = 1, \ldots, p \right\}.$$

That is, $\mathcal{G}$ *generates* K if K is the set of all nonnegative linear combinations of vectors in $\mathcal{G}$. The cones considered here are generated by a finite number of vectors, but cones need not be finitely generated (consider an ice cream cone or a dunce cap).

The *polar cone* of a cone K, denoted by K°, is the cone defined by

$$K^\circ = \left\{ w \mid w^T v \leq 0 \text{ for all } v \in K \right\}.$$

These are the vectors that make an angle of 90° or more with every element of K. An example in $\mathbb{R}^2$ of a cone K and its polar K° is shown in Figure 7.3(a).

Given a convex cone K and any vector v, there is a unique closest point in K to v, called the *projection* of v; denote this projection onto K by either v_K or $P_K(v)$. The polar decomposition [186, 226, 280] says any vector v can be written as the sum of its projection onto a cone and its polar:

$$v = v_K + v_{K^\circ},$$

where $v_K^T v_{K^\circ} = 0$. An example in $\mathbb{R}^2$ is shown in Figure 7.3(b). This is the analog in the geometry of cones of the orthogonal decomposition in linear algebra.

Given a set Ω, every point $x \in \Omega$ has two important cones associated with it: the cone of tangents and the cone of normals. A vector v is *tangent* to Ω at x if, for every sequence x_i in Ω converging to x and positive sequence α_i decreasing to 0, there is a

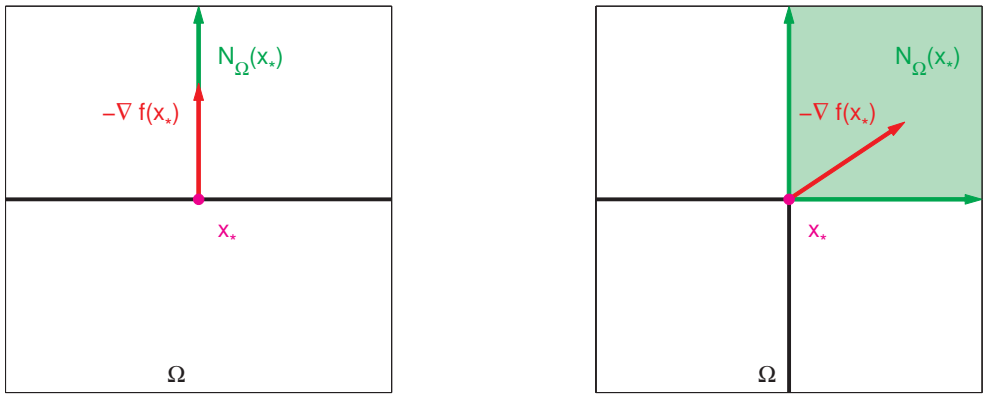


Fig. 7.4 *Two examples of constrained local minimizers.*

sequence v_i converging to v such that $x_i + \alpha_i v_i \in \Omega$. Roughly speaking, a vector v is tangent to Ω at x if one can proceed from x along v for at least some short distance and remain in Ω. A standard result is that the set of all vectors tangent to Ω at x is a convex cone [66]. This cone is called the *(Clarke) tangent cone* and is denoted by $\mathcal{T}_\Omega(x)$. The cone polar to the tangent cone is called the *normal cone* and is denoted by $\mathcal{N}_\Omega(x)$. Again roughly speaking, the normal cone consists of the vectors that point "outward" from Ω at x. Note that if x is in the interior of Ω, then $\mathcal{N}_\Omega(x) = \{0\}$ and $\mathcal{T}_\Omega(x)$ is all of $\mathbb{R}^n$.

7.3. First-Order Necessary Conditions for a Constrained Minimizer. Next, consider the necessary conditions for constrained minimizers. A point $x_* \in \Omega$ is a local constrained minimizer if, for some $r > 0$,

$$f(x_*) \leq f(x) \quad \text{for all } x \in \Omega \text{ such that } \|x - x_*\| < r.$$

If x_* is a local constrained minimizer, then

$$-\nabla f(x_*) \in \mathcal{N}_\Omega(x_*). \tag{7.2}$$

If x_* is a local minimizer of (7.1) and x_* is in the interior of Ω, then $\mathcal{N}_\Omega(x_*) = \{0\}$ and (7.2) says $\nabla f(x_*) = 0$. On the other hand, if x_* is on the boundary of Ω and $\nabla f(x_*) \neq 0$, (7.2) says that the only way to obtain further decrease locally is to leave the feasible region. Two examples for which (7.2) holds are shown in Figure 7.4.

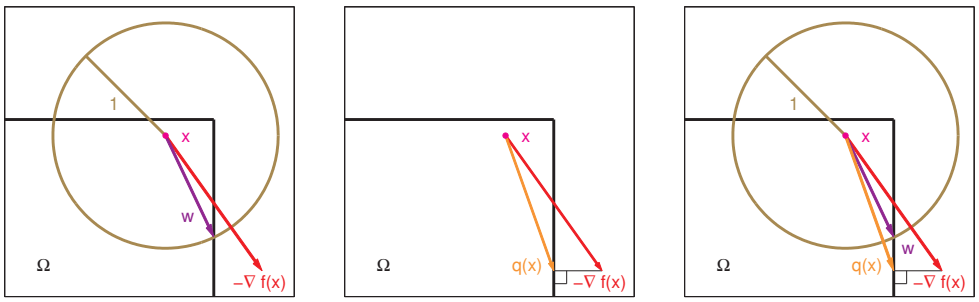
A constraint $c^{(i)}(x)$ in (7.1) is called *binding* at x if $c^{(i)}(x) = 0$. Under various relatively mild conditions called *constraint qualifications*, the cone of vectors normal to Ω at x is the cone generated by the outward-pointing normals to the constraints that are binding at x. (See any of [69, 106, 192, 197] for a discussion of constraint qualifications.)

Let $\mathcal{B}(x)$ denote the indices of the constraints binding at x:

$$\mathcal{B}(x) = \left\{ \; i \; \mid \; c^{(i)}(x) = 0 \; \right\}.$$

If a suitable constraint qualification holds at x, then the normal cone is given by

$$\mathcal{N}_\Omega(x) = \left\{ \; v \; \mid \; v = \sum_{i \in \mathcal{B}(x)} \alpha^{(i)} \nabla c^{(i)}(x), \; \alpha^{(i)} \geq 0 \text{ for all } i \in \mathcal{B}(x) \; \right\}.$$



(a) The w that gives the measure $\chi(x)$.

(b) $q(x)$.

(c) w and $q(x)$ superimposed for comparison.

Fig. 7.5 *The geometry of* $\chi(x)$ *and* $q(x)$.

If the normal cone can be written in this way, one obtains the *Karush–Kuhn–Tucker (KKT)* condition for a constrained local minimizer. The necessary condition $-\nabla f(x_*) \in \mathcal{N}_\Omega(x_*)$ becomes

$$-\nabla f(x_*) = \sum_{i \in \mathcal{B}(x_*)} \alpha^{(i)} \nabla c^{(i)}(x_*), \text{ where } \alpha^{(i)} \geq 0 \text{ for } i \in \mathcal{B}(x_*). \tag{7.3}$$

A point for which (7.3) holds is called a *KKT point*, and the $\alpha^{(i)}$ are called *Lagrange multipliers*.

7.4. Measures of Convergence to KKT Points. The necessary condition (7.2) has the drawback that it does not serve as a continuous measure of constrained stationarity, because the normal cone $\mathcal{N}_\Omega(x)$ does not depend continuously on x at the boundary of the feasible region Ω. For this reason, two measures of stationarity and convergence to KKT points appear in the discussion that follows. These measures of stationarity are widely used in nonlinear programming; they are not peculiar to the analysis of GSS methods.

The first measure is the following. For $x \in \Omega$, let

$$\chi(x) = \max_{\substack{x+w \in \Omega \\ \| w \| \leq 1}} -\nabla f(x)^T w. \tag{7.4}$$

Loosely speaking, $\chi(x)$ captures the degree to which the direction of steepest descent is outward-pointing with respect to the portion of the feasible region near x. The sets defining χ and the maximizer w in (7.4) are depicted in Figure 7.5(a).

When Ω is convex, or otherwise has a geometrically "nice" boundary vis-à-vis the aforementioned constraint qualifications, the function χ has the following properties [69]:

1. $\chi(x)$ is continuous.
2. $\chi(x) \geq 0$.
3. $\chi(x) = 0$ if and only if x is a KKT point for (7.1).

If $\chi(x) = 0$, then there is no direction making an angle of less than 90° with the direction of steepest descent along which one can move for any distance, no matter

how short, and still remain feasible. Showing that $\chi(x_k) \to 0$ as $k \to \infty$ constitutes a global first-order convergence result.

The second measure of constrained stationarity involves projections onto Ω under the assumption that Ω is convex, which is the case for bound and linearly constrained problems. Let $P_\Omega(x)$ denote the point in Ω closest to x, which is well-defined if Ω is convex. Define

$$q(x) = P_\Omega(x - \nabla f(x)) - x.$$

A point x_* is a KKT point of (7.1) if and only if $q(x_*) = 0$ (see [69, 99], for instance). Thus, a global first-order convergence result could also be obtained by showing that $\| q(x_k) \| \to 0$. Like χ, $\| q(x_k) \|$ reflects the extent to which the direction of steepest descent is outward-pointing with respect to the feasible region. Unlike $\chi(x)$, however, $\| q(x_k) \|$ depends on the global geometry of the feasible region, since the projection $P_\Omega(x - \nabla f(x))$ is not necessarily within a predetermined distance of x. Figure 7.5(b) gives a depiction of $q(x)$, and Figure 7.5(c) superimposes Figure 7.5(a) and Figure 7.5(b) to show the differences between $\chi(x)$ and $q(x)$.

In the constrained theory, the quantities $\chi(x)$ and $\| q(x) \|$ play roles similar to that of $\| \nabla f(x) \|$ in the unconstrained theory. Each provides a *continuous* measure of how close a point x is to constrained stationarity, as in the theory of methods based explicitly on derivatives. The presentation to follow relies on the measure $\chi(x)$. Though the measure $\| q(x) \|$ is particularly well suited for the discussion of both derivative-based and direct search algorithms for bound constrained problems (e.g., [52, 67, 69, 99, 167]), the measure $\chi(x)$ is used here in the simpler case of bound constraints to shed some light on the more complicated situation with general linear constraints. The use of $\chi(x)$ is by no means essential, however, and direct search methods for bound constrained problems can be analyzed using $\| q(x) \|$ (e.g., [167]).

8. Direct Search and Constrained Optimization. Direct search methods are usually presented for the case of unconstrained optimization. What is not so widely appreciated is that since the time they first appeared in the 1950s and 1960s, they have been adapted in various ways to handle constrained problems [39, 54, 84, 95, 122, 137, 139, 152, 157, 178, 194, 207, 227, 245].

Early proposals for direct search algorithms for constrained optimization met with some success in practice. However, they are not used much today, unlike their unconstrained counterparts. In fact, the early developers of direct search methods for constrained optimization were aware that their heuristics did not always work in practice. For instance, see Box's comments on Rosenbrock's method in [39] or Keefer's comments on Box's method in [152]. At the time, the emphasis in optimization was on the development of new computational methods rather than on establishing their theoretical properties. This is not surprising. The early development of direct search methods preceded the first global convergence analysis of even unconstrained optimization algorithms using the Armijo–Goldstein–Wolfe conditions (2.4)–(2.5).

Nevertheless, some of the ideas proposed in the early development of direct search methods can be placed on a firm theoretical basis. This is particularly true for bound and linearly constrained problems. The approach outlined here for bound and general linear constraints is a feasible iterates approach [167, 168, 173, 175]. Philosophically, in the presence of linear constraints, the situation is similar to that of the unconstrained case. At each iteration the method must work with a sufficiently rich set of search directions to ensure that, regardless of the direction of steepest descent, one of the search directions is a descent direction if the current iterate is not a KKT point. But,

as discussed in section 7.1 and illustrated in Figure 7.1(c), the presence of constraints introduces a new consideration. Now it is not enough simply to ensure a direction of descent; it must also be possible to take sufficiently long steps along descent directions and remain feasible. This suggests modifying the choice of search directions for GSS methods. The following sections discuss how to do so.

8.1. The Case of Bound Constraints: Compass Search, Again. Bound constraints are treated here apart from general linear constraints for several reasons. Bounds and their geometry are much simpler to describe, as are the modifications needed to tailor GSS methods to account for the constraints, as the illustrations in Figure 7.2 suggest. At the same time, the bound constrained case serves to make clearer the issues that must be addressed for general linear constraints.

The bound constrained problem is

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & \ell \le x \le u. \end{array} \tag{8.1}$$

The inequality $\ell \le x \le u$ is interpreted to hold term by term; i.e., $\ell^{(i)} \le x^{(i)} \le u^{(i)}$, $i = 1, \ldots, n$. We allow the possibility that some of the variables may be unbounded by allowing $\ell^{(i)} = -\infty$ and $u^{(i)} = +\infty$. The feasible region is $\Omega = \{x \,|\, \ell \le x \le u\}$.

Compass search is the simplest example of a globally convergent algorithm for solving (8.1). It may be surprising that this exceedingly simple algorithm is guaranteed to converge to a KKT point of (8.1), but such is the case.

In two dimensions, compass search for bound constraints can be summarized as follows. Try steps to the East, West, North, and South. If one of these steps yields reduction in the value of the objective function *and satisfies the bounds*, then this point becomes the new iterate. If none of these steps yields *feasible* improvement, try again with steps half as long. As is usual in the unconstrained case, the algorithm terminates when the step length falls below a given threshold.

A quick review of the description of compass search found in section 1.1 will reveal that the only difference between the unconstrained and the bound constrained variants of compass search is the insistence that every iterate (including the initial iterate x_0) must be contained in the feasible region—as emphasized in the description above. In other words, $x_k \in \Omega$ for all $k = 0, 1, \ldots$.

The first reference that we can find in the direct search literature which explicitly observes that the coordinate directions should be used as search directions when there are only bound constraints is Keefer's Simpat algorithm [152]. Keefer notes that the pattern associated with the method of Hooke and Jeeves [139] conforms in a natural way to the boundary of the feasible region and is well suited for coping with bounds. The Simpat algorithm combines the use of the Nelder–Mead simplex algorithm [194] in the interior of the feasible region with the use of the Hooke and Jeeves pattern search algorithm near the boundary.

As observed in [167], there is an equivalence between this GSS method for bound constrained problems and an *exact penalization* approach to problem (8.1). Applying the version of compass search just discussed to problem (8.1) produces exactly the same iterates as applying compass search method to the unconstrained problem

$$\text{minimize } \hat{f}(x), \tag{8.2}$$

where

$$\hat{f}(x) = \begin{cases} f(x) & \text{if } x \in \Omega, \\ +\infty & \text{otherwise.} \end{cases} \tag{8.3}$$

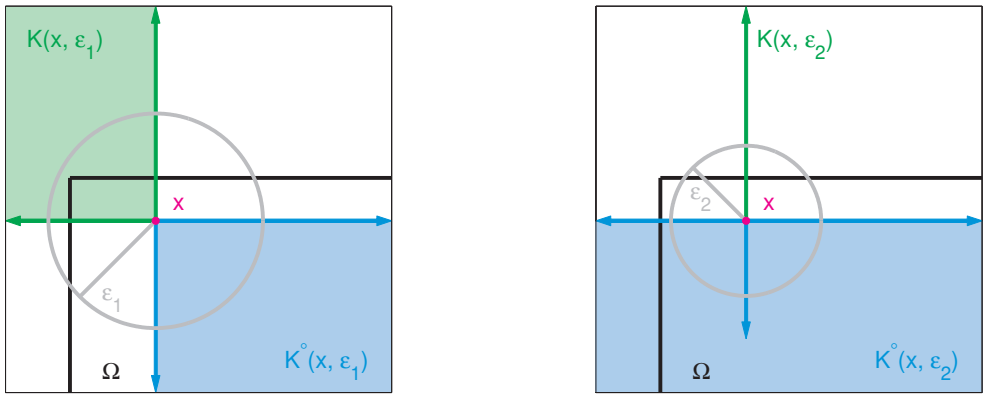


Fig. 8.1 *The cones $K(x,\varepsilon)$ and $K^\circ(x,\varepsilon)$ for the values ε_1 and ε_2.*

In fact, this is one classical approach used with direct search methods to ensure that the iterates produced remain feasible (see, for instance, [152, 194, 245], as well as the comments in section 8.3.3). GSS (and other direct search) methods are directly applicable to (8.2) using the *exact penalty function* (8.3). However, this exact penalization approach should not be applied with an arbitrary generating set $\mathcal{G}$. As noted in section 7.1, the search directions should respect the boundary of the feasible region. The simplest condition on $\mathcal{G}$ that suffices to ensure global convergence is that $\mathcal{G}$ contain $\mathcal{D}_\oplus$ (or scaled variants), as in the case of compass search.

8.1.1. The Geometry of the Nearby Boundary. Let x be an iterate generated by compass search. It is the interaction of the choice of search directions with the portion of the boundary of Ω near x (i.e., the bound constraints that are binding or nearly binding at x) that is of concern. Given $x \in \Omega$, let $K(x,\varepsilon)$ denote the cone generated by 0 and the outward-pointing normals of the constraints within a distance ε of x, which are

$$\{e_i \mid u^{(i)} - x^{(i)} \le \varepsilon\} \cup \{-e_i \mid x^{(i)} - \ell^{(i)} \le \varepsilon\}.$$

The cone $K(x,\varepsilon)$ is generated by the normals to the faces of the boundary within distance ε of x. If $K(x,\varepsilon) = \{0\}$, as is true in the unconstrained case or when x is well within the interior of the feasible region, then $K^\circ(x,\varepsilon) = \mathbb{R}^n$. The significance of $K(x,\varepsilon)$ is that for suitable choices of ε, its polar $K^\circ(x,\varepsilon)$ approximates the feasible region near x. (More precisely, $x + K^\circ(x,\varepsilon)$ approximates the feasible region near x, but we will continue to abuse the language in this manner.)

These cones are illustrated in Figure 8.1 for two values of ε, ε_1 and ε_2. In the left-hand figure, $K(x,\varepsilon_1)$ is the shaded cone between the two green vectors, while $K^\circ(x,\varepsilon_1)$ is the shaded cone between the two blue vectors. In the right-hand figure, $K(x,\varepsilon_2)$ is the green ray, while $K^\circ(x,\varepsilon_2)$ is the shaded region below the blue line. In both cases, $K^\circ(x,\varepsilon)$ mirrors the geometry of the nearby boundary of Ω, where "nearby" is defined in terms of the particular choice of ε. On the other hand, in both cases the actual tangent cone at x is $\mathcal{T}_\Omega(x) = \mathbb{R}^n$, which does not capture the nearby shape of Ω.

The polar cone $K^\circ(x,\varepsilon)$ is important because if $K^\circ(x,\varepsilon) \neq \{0\}$, then the search can proceed from x along all directions in $K^\circ(x,\varepsilon)$ for at least a distance of ε and still

remain inside the feasible region. This is not the case for directions in the tangent cone of the feasible region at x, since the latter cone does not reflect the proximity of the boundary for points close to, but not on, the boundary.

8.1.2. Obtaining Feasible Descent. At every iteration k, the set of search directions $\mathcal{G}_k$ in compass search is $\mathcal{D}_\oplus = \{e_1, e_2, \ldots, e_n, -e_1, -e_2, \ldots, -e_n\}$. Obviously, since these vectors form a positive basis for $\mathbb{R}^n$, one of them is guaranteed to be a descent direction, provided $-\nabla f(x_k) \neq 0$, no matter the orientation of $-\nabla f(x_k)$. However, as discussed in section 7.1, this is not enough. There remains the potential for steps that are excessively small not because there is little feasible improvement possible, but solely because the unhappy orientation of the steps causes them to bump up against the boundary.

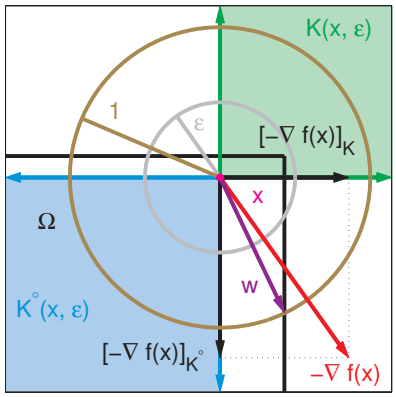
A first step toward showing that compass search produces at least a subsequence of iterates that converges to a KKT point of (8.1) is to show that if the iterate x_k is not a KKT point, then at least one of the search directions is a descent direction along which the search can progress a reasonable distance and remain feasible. The feature of compass search that guarantees a direction of feasible descent for problem (8.1) is that some subset of the set of search directions $\mathcal{D}_\oplus$ necessarily reflects the shape of the feasible region near the current iterate x_k, no matter where x_k lies in Ω. A feasible descent direction will be implicitly identified as a generator of a cone $K^\circ(x, \varepsilon)$ for some $\varepsilon > 0$. As illustrated previously in Figure 8.1, the cone $K^\circ(x, \varepsilon)$ approximates the feasible region near x.

Now, if the algorithm has not arrived at a KKT point (i.e., $\chi(x) \neq 0$), then there must be a descent direction from x along which the search can move for some distance and remain feasible. If ε is not too large, $K^\circ(x, \varepsilon)$ should be a good enough approximation of Ω near x. If $K^\circ(x, \varepsilon)$ is a good enough approximation of Ω near x, then there should also be a descent direction in $K^\circ(x, \varepsilon)$ along which the search can move for a distance of at least ε and remain feasible. Since such a descent direction must be a nonnegative combination of the generators of $K^\circ(x, \varepsilon)$, at least one of these generators should be a descent direction, too. But in the case of bound constraints, $K^\circ(x, \varepsilon)$ necessarily is generated by a subset of $\mathcal{D}_\oplus$, the set of search directions for compass search.

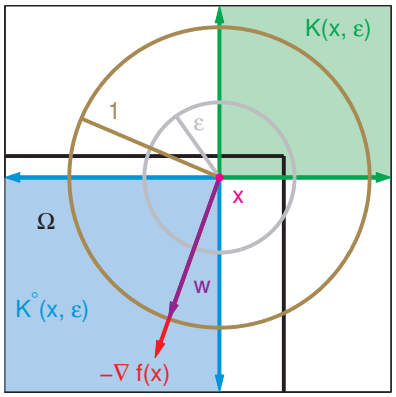
The upshot is that if $\chi(x) \neq 0$, then at least one of the search directions in $\mathcal{D}_\oplus$ will be a descent direction, and the search can take a suitably long step along this direction and remain feasible. The situation is illustrated in Figure 8.2. In these pictures, at least one of the generators of $K^\circ(x, \varepsilon)$, $-e_2$, is a feasible descent direction. Moreover, the search can move a distance at least of the order of ε along this direction and remain feasible, so the issue does not arise of excessively small steps due solely to bumping up against the boundary.

The need for ε to be sufficiently small is illustrated in Figure 8.3(a). Note that x is not a KKT point, but for this particular choice of ε, there is no descent direction in $K^\circ(x, \varepsilon)$. However, if ε were slightly smaller, as shown in Figure 8.3(b), then $K(x, \varepsilon) = \{0\}$ and $K^\circ(x, \varepsilon) = \mathbb{R}^n$. If so, $\mathcal{D}_\oplus$ is the set of generators for $\mathbb{R}^n$ (as in the unconstrained case), at least one of the search directions would be a descent direction (in Figure 8.3(b), either e_1 or e_2), and the search would be able to move a distance of at least ε along this direction and remain feasible. It will turn out that ε must be small relative to $\chi(x)$.

For compass search, the simplicity of the boundary of Ω for the bound constrained problem (8.1) means that $\mathcal{D}_\oplus$ automatically contains all the generators for any possible

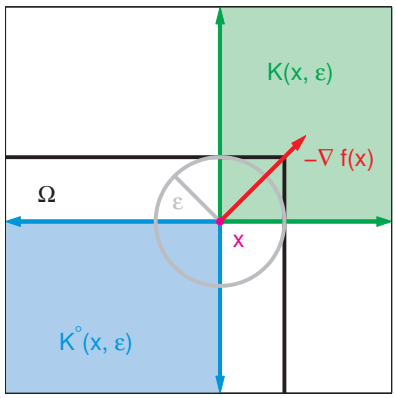


(a) When $-\nabla f(x) \notin K^\circ(x,\varepsilon)$ and its projection onto $K^\circ(x,\varepsilon)$ is not equal to zero.

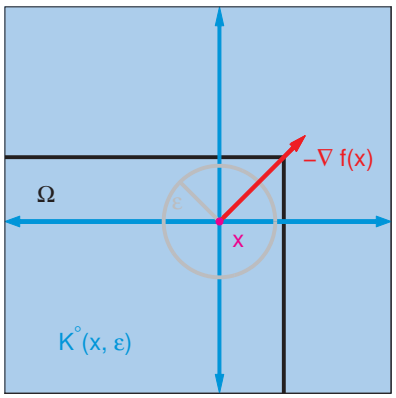


(b) When $-\nabla f(x) \in K^\circ(x,\varepsilon)$.

Fig. 8.2 *At least one generator of $K^\circ(x,\varepsilon)$ that yields feasible descent for a step of at least length ε.*



(a) When ε is not sufficiently small.



(b) When ε is sufficiently small.

Fig. 8.3 *Why ε must be small enough to ensure feasible descent along at least one generator of $K^\circ(x,\varepsilon)$ for a step of at least length ε.*

$K^\circ(x,\varepsilon)$. In this way, compass search is able to pick up any feasible improvement in the objective without knowing the direction of steepest descent.

The next proposition says that if the projection $[-\nabla f(x)]_{K^\circ}$ is nonzero, then one of the generators for $K^\circ(x,\varepsilon)$ is a descent direction. This is an analog of (3.2). In this case, however, the set of search directions $\mathcal{G}_{K^\circ}$ is the subset of $\mathcal{G} = \mathcal{D}_\oplus$ that generates $K^\circ(x,\varepsilon)$, rather than $\mathbb{R}^n$. As a result, the bound is in terms of the projection of the vector $-\nabla f(x)$ into $K^\circ(x,\varepsilon)$.

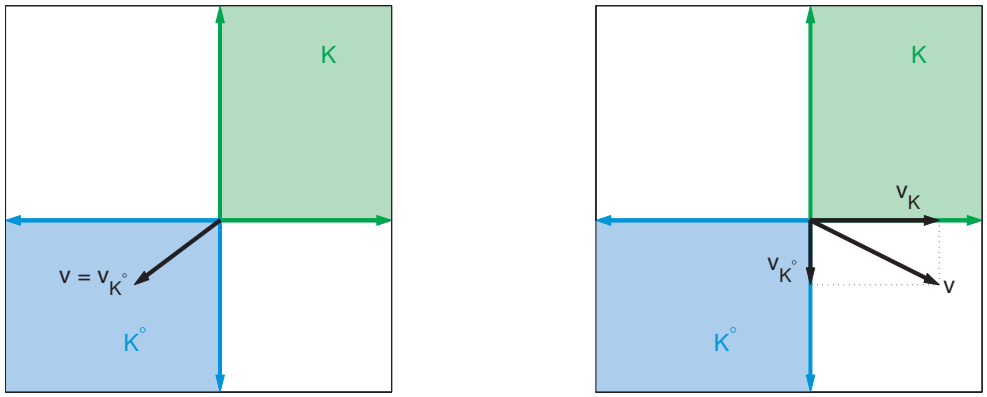


Fig. 8.4 *The geometry of Proposition* 8.1.

Proposition 8.1 is illustrated in Figure 8.4 for a generic vector v. In both situations, the projection v_{K° makes an angle of less than 90° with at least one of the generators of $K^\circ(x,\varepsilon)$. (Be warned, though, that the two-dimensional case is a bit misleading since the projection v_{K° is collinear with one of the generators, while in higher dimensions the projection may lie in the interior of a face.)

PROPOSITION 8.1. *Let $\mathcal{G} = \mathcal{D}_\oplus$. Let $x \in \Omega$ and $\varepsilon \geq 0$, and let $K^\circ = K^\circ(x,\varepsilon)$ and $K = K(x,\varepsilon)$ for the bound constrained problem* (8.1). *Let $\mathcal{G}_{K^\circ} \subseteq \mathcal{G}$ be the set of generators of K°. Then, if $[-\nabla f(x)]_{K^\circ} \neq 0$, there exists $d \in \mathcal{G}_{K^\circ}$ such that*

$$\frac{1}{\sqrt{n}} \| [-\nabla f(x)]_{K^\circ} \| \leq -\nabla f(x)^T d.$$

Proof. Let $v = -\nabla f(x)$. The polar decomposition of v gives $v = v_{K^\circ} + v_K$. In this case,

$$v_{K^\circ}^{(j)} = \begin{cases} v^{(j)} & \text{if } v^{(j)} > 0 \quad \text{and} \quad e_j \in K^\circ, \\ v^{(j)} & \text{if } v^{(j)} < 0 \quad \text{and} \quad -e_j \in K^\circ, \\ 0 & \text{otherwise} \end{cases}$$

and

$$v_K^{(j)} = \begin{cases} v^{(j)} & \text{if } v^{(j)} > 0 \quad \text{and} \quad e_j \in K, \\ v^{(j)} & \text{if } v^{(j)} < 0 \quad \text{and} \quad -e_j \in K, \\ 0 & \text{otherwise.} \end{cases}$$

Let j denote the index of the largest magnitude element of v_{K°; i.e.,

$$| v_{K^\circ}^{(j)} | = \max_i | v_{K^\circ}^{(i)} |.$$

If $d = \text{sign}(v_{K^\circ}^{(j)}) e_j$, then

$$v_{K^\circ}^T d = | v_{K^\circ}^{(j)} | \geq \frac{\| v_{K^\circ} \|}{\sqrt{n}}.$$

Since $|v_{K^\circ}^{(j)}| > 0$ by the assumption that $v_{K^\circ} \neq 0$, the vector $\text{sign}(v_{K^\circ}^{(j)})\, e_j$ is necessarily a generator for K°. Furthermore, $v_K^{(j)} = 0$, so $v_K^T d = 0$. Thus,

$$\frac{1}{\sqrt{n}} \| v_{K^\circ} \| \leq (v_{K^\circ} + v_K)^T d = v^T d. \qquad \square$$

The next result, Proposition 8.2, relates $\chi(x)$ to the descent direction guaranteed by Proposition 8.1.

PROPOSITION 8.2. *Let $\mathcal{G} = \mathcal{D}_\oplus$. Let $x \in \Omega$ and $\varepsilon \geq 0$, and let $K^\circ = K^\circ(x, \varepsilon)$ and $K = K(x, \varepsilon)$ for the bound constrained problem* (8.1). *Then*

$$\chi(x) \leq \| [-\nabla f(x)]_{K^\circ} \| + \sqrt{n}\, \| [-\nabla f(x)]_K \|\, \varepsilon. \tag{8.4}$$

Proof. The proof relies on the fact that one can move along a direction in $K(x, \varepsilon)$ for a distance of at most $\sqrt{n}\varepsilon$ before going infeasible, as illustrated in Figure 8.3(a).

Let $v = -\nabla f(x)$. Applying the polar decomposition $v = v_K + v_{K^\circ}$ to (7.4) yields

$$\chi(x) \leq \max_{\substack{x+w \in \Omega \\ \| w \| \leq 1}} v_{K^\circ}^T w + \max_{\substack{x+w \in \Omega \\ \| w \| \leq 1}} v_K^T w. \tag{8.5}$$

The first term on the right is easily bounded as

$$\max_{\substack{x+w \in \Omega \\ \| w \| \leq 1}} v_{K^\circ}^T w \leq \| v_{K^\circ} \|.$$

Meanwhile,

$$\max_{\substack{x+w \in \Omega \\ \| w \| \leq 1}} v_K^T w = \max_{\substack{x+w \in \Omega \\ \| w \| \leq 1}} v_K^T (w_K + w_{K^\circ}) \leq \max_{\substack{x+w \in \Omega \\ \| w \| \leq 1}} v_K^T w_K,$$

since $v_K^T w_{K^\circ} \leq 0$ for any w. However, the definition of $K(x, \varepsilon)$ ensures that if $x + w \in \Omega$, no component of w_K can be larger than ε (see Figure 8.3(a)). Therefore, $\| w_K \| \leq \sqrt{n}\varepsilon$, and (8.5) yields

$$\chi(x) \leq \| v_{K^\circ} \| + \sqrt{n} \| v_K \| \varepsilon.$$

Note that if $K = \{0\}$, then $K^\circ = \mathbb{R}^n$. $\square$

To better understand the bound (8.4), suppose that x is not a KKT point, so $\chi(x) > 0$. Suppose also that $\| -\nabla f(x) \| \leq \gamma$ for some γ. Then $\| [-\nabla f(x)]_K \| \leq \gamma$, as well. If ε is sufficiently small compared to $\chi(x)$, say, $\sqrt{n}\gamma\varepsilon = \frac{1}{2}\chi(x)$, then Proposition 8.2 yields $\chi(x) \leq 2 \| [-\nabla f(x)]_{K^\circ} \|$. Thus, Proposition 8.2 says that if $\chi(x) \neq 0$, then $[-\nabla f(x)]_{K^\circ} \neq 0$, provided ε is sufficiently small compared to $\chi(x)$.

However, Proposition 8.1 then guarantees that one of the generators of $K^\circ(x, \varepsilon)$ is a descent direction. Moreover, the search can move along this direction at distance of at least ε and remain feasible, and ε is of roughly the same order as $\chi(x)$. This means that one of the search directions is a feasible descent direction, and, in addition, the problem of excessively small steps along this direction due solely to the proximity of the boundary does not arise.

8.1.3. A Bound on Stationarity at Unsuccessful Iterations. The following result, analogous to Theorem 3.3 in the unconstrained case, bounds $\chi(x_k)$ in terms of the value of the step-length control parameter Δ_k at unsuccessful iterations.

THEOREM 8.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable, and suppose ∇f is Lipschitz continuous with constant M and $\| \nabla f(x) \| \leq \gamma$ for all $x \in \Omega$. Then the bound constrained version of compass search produces iterates such that for any $k \in \mathcal{U}$,*

$$\chi(x_k) \leq \sqrt{n}(M + \gamma)\Delta_k.$$

Proof. There are two cases to consider.

Case I. Let $\text{int}(\Omega)$ denote the interior of Ω. If, for all $d \in \mathcal{G}$, $x_k + \Delta_k d \notin \text{int}(\Omega)$, then every trial point is either on the boundary of or outside Ω; i.e., for $i = 1, \ldots, n$,

$$x_k^{(i)} - \Delta_k \leq \ell^{(i)} \quad \text{and} \quad x_k^{(i)} + \Delta_k \geq u^{(i)}.$$

This means that if $x + w \in \Omega$, no component of w can be larger than Δ_k, so the length of the longest vector w for which $x + w \in \Omega$ cannot exceed $\sqrt{n}\Delta_k$. Thus,

$$\chi(x_k) = \max_{\substack{x_k + w \in \Omega \\ \| w \| \leq 1}} -\nabla f(x_k)^T w \leq \max_{\substack{x_k + w \in \Omega \\ \| w \| \leq 1}} \| \nabla f(x_k) \| \, \| w \| \leq \sqrt{n}\Delta_k \| \nabla f(x_k) \| \leq \sqrt{n}\gamma\Delta_k.$$

Case II. Otherwise, there exists at least one $d \in \mathcal{G}$ such that $x_k + \Delta_k d \in \text{int}(\Omega)$. In fact, the cone $K^\circ = K^\circ(x_k, \Delta_k)$ is generated by the set $d \in \mathcal{G}$ such that $x_k + \Delta_k d \in \text{int}(\Omega)$. In this case, applying Proposition 8.2 yields

$$\begin{aligned} \chi(x_k) &\leq \| \, [-\nabla f(x_k)]_{K^\circ} \, \| + \sqrt{n} \, \| \, [-\nabla f(x_k)]_K \, \| \, \Delta_k \\ &\leq \| \, [-\nabla f(x_k)]_{K^\circ} \, \| + \sqrt{n}\gamma\Delta_k. \end{aligned}$$

Following the same reasoning as in the proof of Theorem 3.3 with the first equation replaced by the result of Proposition 8.1 yields

$$\| \, [-\nabla f(x_k)]_{K^\circ} \, \| \leq \sqrt{n} M \Delta_k.$$

The theorem follows. □

Theorem 8.3 means that the other element needed to prove that at least one limit point of the subsequence of unsuccessful iterations is a KKT point is showing that

$$\liminf_{\substack{k \to \infty \\ k \in \mathcal{U}}} \Delta_k = 0. \tag{8.6}$$

However, in the case of compass search this follows from the arguments already seen in section 3.7.2.

The analysis for general linear constraints will follow a similar pattern. The suitable choice of search directions and results on the geometry of the feasible region first lead to a bound on $\chi(x_k)$ in terms of Δ_k. The globalization strategies already seen for GSS algorithms for unconstrained problems then yield (8.6) with little or no alteration.

The reader may have noticed that ε plays only a minor role in the analysis for the bound constrained case. The use of $\| \, q(x) \, \|$, rather than $\chi(x)$, in the analysis of the bound constrained case would obviate the need to introduce $K(x, \varepsilon)$. The additional machinery was introduced here only to illustrate some of the issues that now become of substantive concern in the consideration of general linear constraints.

8.2. Direct Search Methods for General Linear Constraints. Next consider the general linearly constrained problem, written as

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && Ax \leq b. \end{aligned} \tag{8.7}$$

The feasible set is $\Omega = \{ \, x \mid Ax \leq b \, \}$.

As with bound constrained problems, the set of search directions $\mathcal{D}_k$ for linearly constrained optimization should reflect the geometry of any portion of the boundary of the feasible region near x_k. However, the feasible region is now a general polyhedron

rather than a box. This more complex geometry makes the description of the search directions more complicated, as discussed next.

For derivative-free methods, the basic principles discussed here for handling the geometry of linear constraints were first considered by May [178] in his extension to linearly constrained problems of Mifflin's algorithm [181]. May's approach is not a pure direct search method since it relies, instead, on coarse finite-difference approximations to the derivative of f.

8.2.1. The Geometry of the Nearby Boundary. Once again, the focus is on that portion of the boundary near x. Unlike the bound constrained case, though, the boundary of the polyhedron Ω is no longer so regular. Further, trying to assemble a single generating set $\mathcal{G}$ that can accommodate all possible configurations is a combinatorial problem, equivalent to vertex enumeration [14], as discussed further in section 8.2.8.

Let a_i^T be the ith row of the constraint matrix A in (8.7). Given $x \in \Omega$ and $\varepsilon \geq 0$, define the index set $I(x, \varepsilon)$ by

$$i \in I(x, \varepsilon) \text{ if } x \text{ is within distance } \varepsilon \text{ of the set } \left\{ y \mid a_i^T y = b^{(i)} \right\}.$$

For $i \in I(x, \varepsilon)$, define

$$\nu_i(x, \varepsilon) = a_i. \tag{8.8}$$

These are the outward-pointing normals to the corresponding faces of Ω. Examples are shown in Figure 8.5 for two different choices of $x \in \Omega$.

Given $x \in \Omega$, $K(x, \varepsilon)$ denotes the cone generated by 0 and the vectors $\nu_i(x, \varepsilon)$ for $i \in I(x, \varepsilon)$. The cone $K(x, \varepsilon)$ is thus generated by the normals to the faces of the boundary within distance ε of x. If $K(x, \varepsilon) = \{0\}$, then $K^\circ(x, \varepsilon) = \mathbb{R}^n$. Observe that $K(x, 0) = \mathcal{N}_\Omega(x)$, while $K^\circ(x, 0) = \mathcal{T}_\Omega(x)$. The cones $K(x, \varepsilon)$ and $K^\circ(x, \varepsilon)$ are illustrated in Figure 8.6 for two different values of ε. In the left-hand figure, $K(x, \varepsilon_1)$ is the shaded cone between the two green vectors, while $K^\circ(x, \varepsilon_1)$ is the shaded cone between the two blue vectors. In the right-hand figure, $K(x, \varepsilon_2)$ is the green ray, while $K^\circ(x, \varepsilon_2)$ is the shaded region below the blue line.

As in the bound constrained case, the polar cone $K^\circ(x, \varepsilon)$ is important because if $\varepsilon > 0$ is sufficiently small, the search can proceed from x along all directions in $K^\circ(x, \varepsilon)$ for a distance ε and still remain inside the feasible region (see Proposition 8.9). This is not the case for directions in $\mathcal{T}_\Omega(x)$, since the tangent cone does not reflect the proximity of the boundary for points close to, but not on, the boundary.

A depiction of $K(x, \varepsilon)$ and $K^\circ(x, \varepsilon)$ in three dimensions is given in Figure 8.7. The point x is near the apex of a right-angle pyramid, and ε is large enough that x is within distance ε of the four faces that meet at the apex. The cone $K(x, \varepsilon)$ is generated by the normals to these four faces.

8.2.2. Specifying the Generating Set. For linearly constrained problems, two geometrical requirements on the set of search directions $\mathcal{D}_k$ have appeared in the literature.

CONDITION 8.4 (see [168, 175]; see also [178] for a special case). *$\mathcal{D}_k$ includes a set $\mathcal{G}_k$ of generators for all of the cones $K^\circ(x_k, \varepsilon)$, $0 \leq \varepsilon \leq \varepsilon^*$, for some $\varepsilon^* > 0$ that is independent of k.*

Note that as ε varies from 0 to ε^* there is only a finite number of distinct cones $K(x_k, \varepsilon)$ since there is only a finite number of faces of Ω. Thus there is only a finite

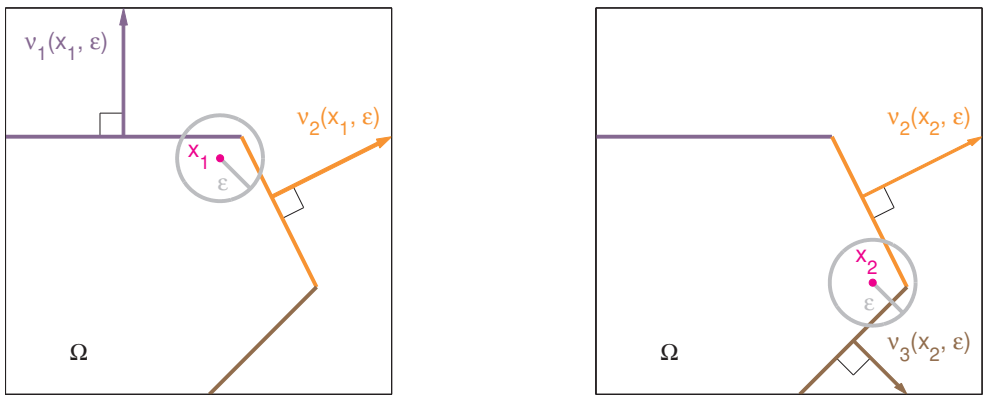


Fig. 8.5 *The outward-pointing normals $\nu_i(x_1, \varepsilon)$ for the index set $I(x_1, \varepsilon) = \{1, 2\}$ and $\nu_i(x_2, \varepsilon)$ for the index set $I(x_2, \varepsilon) = \{2, 3\}$.*

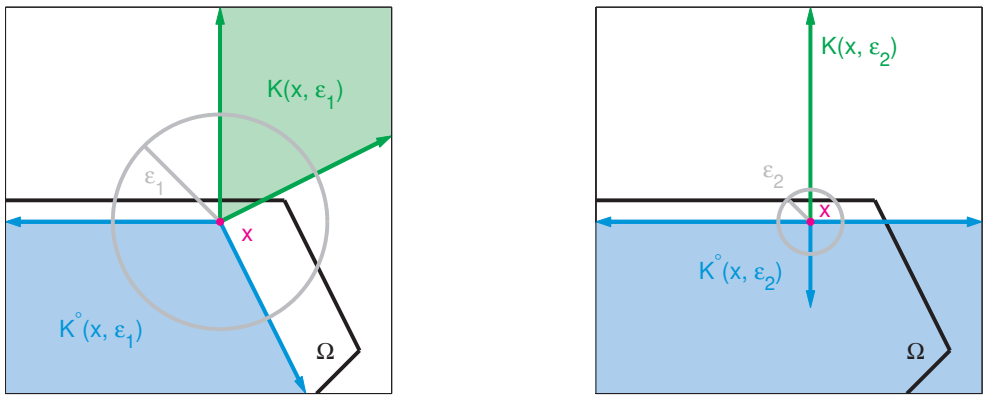


Fig. 8.6 *The cones $K(x, \varepsilon)$ and $K^\circ(x, \varepsilon)$ for the values ε_1 and ε_2.*

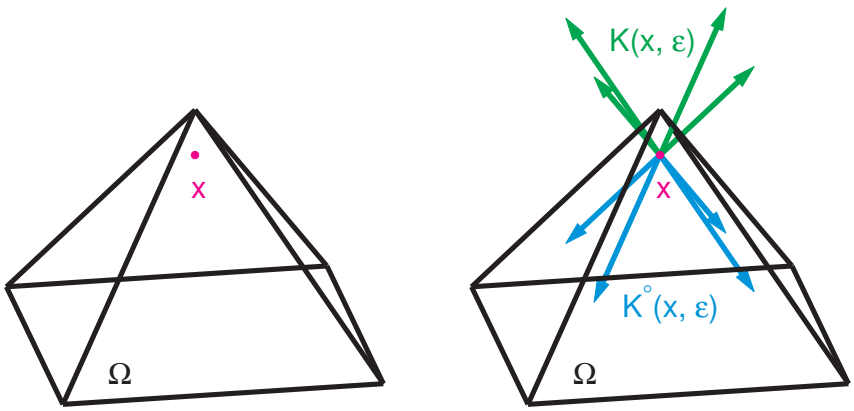


Fig. 8.7 *A pyramid-shaped feasible region with a point x near the apex of the pyramid.*

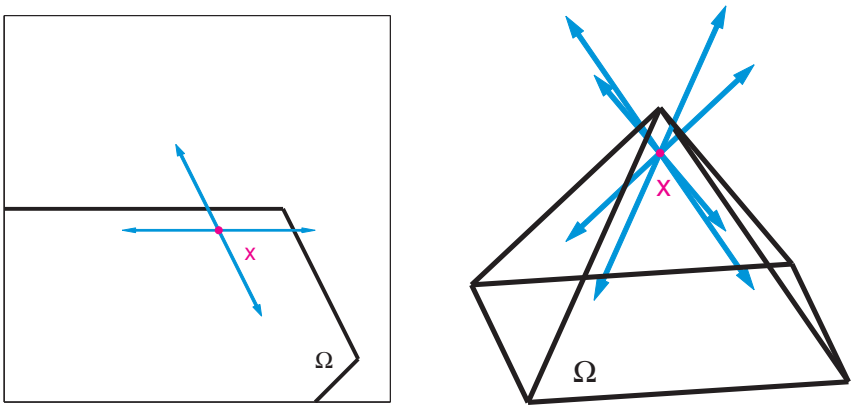


Fig. 8.8 *Sets of search directions that conform to the nearby boundaries.*

number of cones $K^\circ(x_k, \varepsilon)$ as ε varies from 0 to ε^*. For instance, in Figure 8.6, for $0 \le \varepsilon \le \varepsilon_1$ there are only three distinct cones $K(x_k, \varepsilon)$.

CONDITION 8.5 (see [175]). *For $\varepsilon_k > 0$, $\mathcal{D}_k$ includes a set $\mathcal{G}_k$ of generators for $K^\circ(x_k, \varepsilon_k)$.*

The latter condition can be viewed as a way to try to defer looking at some of the search directions that arise in Condition 8.4. The rules for choosing and updating ε_k are discussed later.

The set of search directions must also satisfy an analog of the uniformity condition (3.11). This condition is given in Condition 8.8, after the necessary technical development.

These are only the minimal requirements on the set of search directions. One might also choose to include the generators of $K(x, \varepsilon)$. Since $K(x, \varepsilon) + K^\circ(x, \varepsilon) = \mathbb{R}^n$, one would then have sets of search directions that would capture descent in any direction. The $2n$ search directions $\pm e^{(i)}$, $i = 1, \ldots, n$, in compass search have this property. The question of computing generators of cones is addressed in section 8.2.8.

The name *generating set search* derives from the requirement that $\mathcal{D}_k$ contain generators for the cones $K^\circ(x_k, \varepsilon)$. In the unconstrained case, $K(x_k, \varepsilon) = \{0\}$ and $K^\circ(x_k, \varepsilon) = \mathbb{R}^n$, so $\mathcal{D}_k$ must contain a set of generators for $\mathbb{R}^n$. As noted in section 3.4, a set of generators for $\mathbb{R}^n$ is also called a positive spanning set. In the constrained case, if x_k is away from the boundary, in the sense that $K(x_k, \varepsilon) = \{0\}$, then $K^\circ(x_k, \varepsilon) = \mathbb{R}^n$ and a set of generators for $K^\circ(x_k, \varepsilon)$ is simply a positive spanning set for $\mathbb{R}^n$, as in the unconstrained case. This is illustrated in Figure 8.3(b) for the bound constrained case. Thus, if the iterate is far enough from the boundary of Ω, the set of search directions can look like one for unconstrained optimization, as it should. On the other hand, if x_k is near the boundary, $K(x_k, \varepsilon) \neq \{0\}$ and the generating set must conform to the local geometry of the boundary, as depicted in Figure 8.8.

8.2.3. GSS Algorithms for Linearly Constrained Problems. Besides the specification of the set of search directions, the other major change from Algorithm 3.2 for unconstrained minimization is the way feasibility is treated. As in the bound constrained case, the algorithm is required to begin with a feasible iterate $x_0 \in \Omega$ and thereafter accept only feasible iterates.

Two flavors of linearly constrained GSS algorithms are outlined in Algorithms 8.1 and 8.2. The significant differences with the unconstrained Algorithm 3.2 are colored red.

Algorithm 8.1 most resembles the linearly constrained algorithm described in [168], though here the option of a sufficient decrease step acceptance criterion has been added. The set of search directions $\mathcal{D}_k$ satisfies Condition 8.4. Imposing Condition 8.4 means that the algorithm looks for improvement at feasible points along the edges of cones, some of which are guaranteed to look like the feasible region near x_k. As in the unconstrained case, it is possible (and frequently is the case) that an acceptable step is found without having to look along every potential search direction.

Algorithm 8.2 captures elements of Algorithms 1 and 2 in [175]. It is based on a sufficient decrease criterion, and the search directions satisfy either Condition 8.4 or Condition 8.5. In Step 2, if a full-length step along an essential search direction is infeasible, then one tries a step to the boundary of the feasible region along this direction.

In both Algorithm 8.1 and Algorithm 8.2, if the search directions satisfy Condition 8.4, the failure to find improvement can only be due to the step length, so the response to an unsuccessful iteration is to reduce Δ_k. In Algorithm 8.2, under Condition 8.5, the ε_k used to identify the outward-pointing normals is allowed to vary. So in Algorithm 8.2, an unsuccessful iteration may be due to the step length, but it may also be due to the choice of search directions (or both). The response to an unsuccessful iteration is therefore to reduce both Δ_k and ε_k.

Not all possible variants are considered here. For instance, one could also use truncation to feasible lattice points in connection with a simple decrease criterion for accepting steps. One could also have different step-length control parameters for different directions.

8.2.4. Existence of a Feasible Descent Direction among the Search Directions. There are two facts that one must establish to prove convergence for linearly constrained GSS. First, one must show that one of the search directions is a descent direction, as in the unconstrained and bound constrained cases. Second—and this is a complication due to the constraints—one must show that the search can take a suitably long step along this descent direction and remain feasible. This latter point first was discussed in section 7.1 in connection with Figure 7.1 and then developed in section 8.1.2 for the case of bound constraints.

Proving the existence of a descent direction in $\mathcal{D}_k$ is more complicated than in the unconstrained and bound constrained cases. The following is a crucial geometric fact about the feasible region. For $x \in \Omega$ and $v \in \mathbb{R}^n$, define

$$\hat{\chi}(x; v) = \max_{\substack{x+w \in \Omega \\ \| w \| \leq 1}} v^T w;$$

then $\chi(x) = \hat{\chi}(x; -\nabla f(x))$. The introduction of $\hat{\chi}$ makes clear that the following proposition is a fact about polyhedra, independent of the objective f. It relates $\hat{\chi}(x; v)$ to the projection of v onto $K^\circ(x, \varepsilon)$, as in Proposition 8.2 for bound constraints. Once again, the cone $K^\circ(x, \varepsilon)$ may be viewed as an approximation of the part of the feasible set near x. The proof may be found in [160].

PROPOSITION 8.6. *There exists $c_{8.6} > 0$, depending only on A (the matrix of constraints), such that if $x \in \Omega$, then*

$$\hat{\chi}(x; v) \leq \| P_{K^\circ(x,\varepsilon)}(v) \| + c_{8.6} \| P_{K(x,\varepsilon)}(v) \| \varepsilon.$$

Initialization.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be given. Let Ω define the feasible region.

Let $x_0 \in \Omega$ be the initial guess.

Let $\Delta_{\text{tol}} > 0$ be the step-length convergence tolerance.

Let $\Delta_0 > \Delta_{\text{tol}}$ be the initial value of the step-length control parameter.

Let $\phi_{\max} \geq 1$ be an upper bound on the expansion factor.

Let $\theta_{\max} < 1$ be an upper bound on the contraction parameter.

Let ρ be a scalar function satisfying either (3.8) or (3.9). The choice $\rho \equiv 0$ is acceptable.

Let $\beta_{\max} \geq \beta_{\min} > 0$ be upper and lower bounds, respectively, on the lengths of the vectors in any generating set.

Let $\varepsilon^* > 0$ be the tolerance for choosing the locally active constraints.

Algorithm. For each iteration $k = 1, 2, \ldots$

Step 1. Let $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$, where $\mathcal{G}_k$ satisfies Condition 8.4 and Condition 8.8, $\beta_{\min} \leq \|d\| \leq \beta_{\max}$ for all $d \in \mathcal{G}_k$, and $\mathcal{H}_k$ is a finite (possibly empty) set of additional search directions such that $\beta_{\min} \leq \|d\|$ for all $d \in \mathcal{H}_k$.

Step 2. If there exists $d_k \in \mathcal{D}_k$ such that $f(x_k + \Delta_k d_k) < f(x_k) - \rho(\Delta_k)$ and $x_k + \Delta_k d_k \in \Omega$, then do the following:

- Set $x_{k+1} = x_k + \Delta_k d_k$ (change the iterate).
- Set $\Delta_{k+1} = \phi_k \Delta_k$, where $1 \leq \phi_k \leq \phi_{\max}$ (optionally expand the step-length control parameter).

Step 3. Otherwise, either $x_k + \Delta_k d \notin \Omega$ or $f(x_k + \Delta_k d) \geq f(x_k) - \rho(\Delta_k)$ for all $d \in \mathcal{D}_k$, so do the following:

- Set $x_{k+1} = x_k$ (no change to the iterate).
- Set $\Delta_{k+1} = \theta_k \Delta_k$ where $0 < \theta_k < \theta_{\max} < 1$ (contract the step-length control parameter).
- If $\Delta_{k+1} < \Delta_{\text{tol}}$, then **terminate**.

Algorithm 8.1 *One variant of a linearly constrained GSS method.*

From Proposition 8.6 the following result can be derived. It says that if x is not a KKT point, then one of the generators of $K^\circ(x, \varepsilon)$ is a descent direction for f, provided ε is sufficiently small. For the proof, the reader is referred to [160].

The geometry of Proposition 8.7 is depicted in Figure 8.9. In Figure 8.9(a), there is no descent direction in $K^\circ(x, \varepsilon_1)$, but in Figure 8.9(b), ε_2 is smaller and one edge of $K^\circ(x, \varepsilon_2)$ is a descent direction. See also Figure 8.3 for an example in the case of bound constraints.

PROPOSITION 8.7. *Let $\gamma > 0$ be given. Then there exists $r_{8.7} > 0$, depending only on γ and A (the matrix of constraints), for which the following holds. Suppose that $x \in \Omega$, $\| \nabla f(x) \| < \gamma$, and $\chi(x) > 0$. Given $\varepsilon \geq 0$, let $K^\circ = K^\circ(x, \varepsilon)$, and let*

Initialization.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be given. Let Ω define the feasible region.

Let $x_0 \in \Omega$ be the initial guess.

Let $\Delta_{\text{tol}} > 0$ be the step-length convergence tolerance.

Let $\Delta_0 > \Delta_{\text{tol}}$ be the initial value of the step-length control parameter.

Let $\phi_{\max} \geq 1$ be an upper bound on the expansion factor.

Let $\theta_{\max} < 1$ be an upper bound on the contraction parameter.

Let ρ be a scalar function satisfying (3.9). This time, $\rho \equiv 0$ is not allowed.

Let $\beta_{\max} \geq \beta_{\min} > 0$ be upper and lower bounds, respectively, on the lengths of the vectors in any generating set.

If the search directions satisfy Condition 8.5:

- Let $\varepsilon_0 > 0$ be an initial tolerance for choosing the locally active constraints.
- Let θ_ε, $0 < \theta_\varepsilon < 1$, be a reduction factor for ε_k.

Algorithm. For each iteration $k = 1, 2, \ldots$

Step 1. Let $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$, where $\mathcal{G}_k$ satisfies either Condition 8.4 or Condition 8.5, and Condition 8.8, $\beta_{\min} \leq \|d\| \leq \beta_{\max}$ for all $d \in \mathcal{G}_k$, and $\mathcal{H}_k$ is a finite (possibly empty) set of additional search directions such that $\beta_{\min} \leq \|d\|$ for all $d \in \mathcal{H}_k$.

Step 2. For $d_k \in \mathcal{D}_k$, let $\alpha_k(d)$ be the largest nonnegative value of α for which $x_k + \alpha d \in \Omega$. Let $\bar{\Delta}_k(d) = \min(\Delta_k, \alpha_k(d))$.

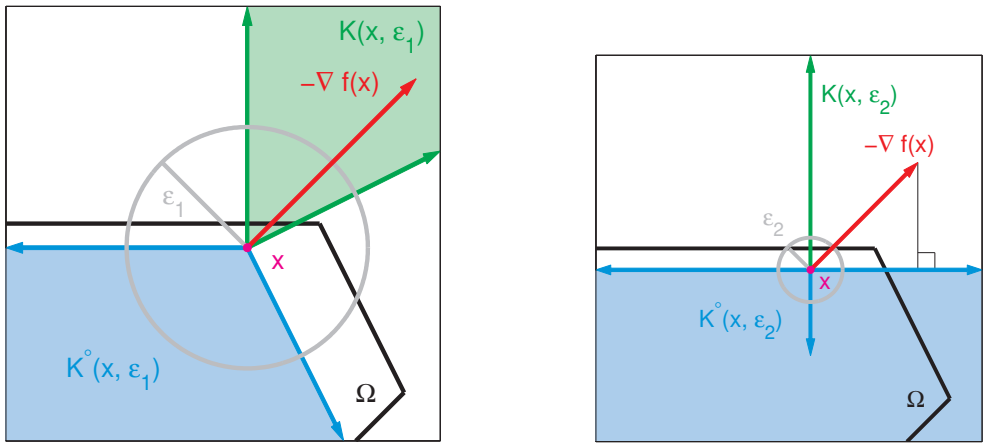
Step 3. If there exists $d_k \in \mathcal{D}_k$ such that $f(x_k + \bar{\Delta}_k(d_k)\, d_k) < f(x_k) - \rho(\bar{\Delta}_k(d_k))$, then do the following:

- Set $x_{k+1} = x_k + \bar{\Delta}_k(d_k)\, d_k$ (change the iterate).
- Set $\Delta_{k+1} = \phi_k \Delta_k$, where $1 \leq \phi_k \leq \phi_{\max}$ (optionally, increase Δ_k).

Step 4. Otherwise, $f(x_k + \bar{\Delta}_k(d)\, d) \geq f(x_k) - \rho(\bar{\Delta}_k(d))$ for all $d \in \mathcal{D}_k$, so do the following:

- Set $x_{k+1} = x_k$ (no change to the iterate).
- Set $\Delta_{k+1} = \theta_k \Delta_k$ where $0 < \theta_k < \theta_{\max} < 1$ (reduce Δ_k).
- If the search directions satisfy Condition 8.5, set $\varepsilon_{k+1} = \theta_\varepsilon \varepsilon_k$ (reduce ε_k).
- If $\Delta_{k+1} < \Delta_{\text{tol}}$, then **terminate**.

Algorithm 8.2 *A second variant of a linearly constrained GSS method.*



(a) No descent direction in $K^\circ(x, \varepsilon_1)$.

(b) A generator of $K^\circ(x, \varepsilon_2)$ is a descent direction.

Fig. 8.9 *$K(x, \varepsilon)$ contains a descent direction if ε is sufficiently small.*

$\mathcal{G}$ be a set of generators of K°. Then there exists $c_{8.7}(\mathcal{G}) > 0$, depending only on $\mathcal{G}$, such that if $\varepsilon < r_{8.7}\,\chi(x)$, then for some $d \in \mathcal{G}$,

$$c_{8.7}(\mathcal{G})\ \chi(x)\ \|\,d\,\| \le -\nabla f(x)^T d. \tag{8.9}$$

Proposition 8.7 means that under Condition 8.4, at least one of the search directions in $\mathcal{D}_k$, which generate $K^\circ(x_k, \varepsilon)$ for all $0 \le \varepsilon \le \varepsilon^*$, is a descent direction for f at x_k. The need for ε to be sufficiently small, compared to χ, is the reason for the requirement in Condition 8.4 that $\mathcal{D}_k$ contain generators for the cones $K^\circ(x_k, \varepsilon)$ for *all* $0 \le \varepsilon \le \varepsilon^*$. In compass search, Condition 8.4 is satisfied automatically by the choice $\mathcal{G} = \mathcal{D}_\oplus$ since then the generators for all possible $K^\circ(x_k, \varepsilon)$ are included automatically. Under Condition 8.5 there is a descent direction in $\mathcal{D}_k$ if ε_k is sufficiently small relative to $\chi(x_k)$.

The bound (8.9) generalizes (3.11) from the unconstrained case. In the unconstrained case, $\Omega = \mathbb{R}^n$, and

$$\chi(x) = \max_{\|\,w\,\| \le 1} -\nabla f(x)^T w = \|\,\nabla f(x)\,\|.$$

Thus (8.9) reduces to

$$c_{8.7}(\mathcal{G})\ \|\,\nabla f(x)\,\|\ \|\,d\,\| \le -\nabla f(x)^T d,$$

which is (3.11), with $c_{8.7}(\mathcal{G}) = \kappa(\mathcal{G})$.

Because $c_{8.7}(\mathcal{G})$ plays the role in the linearly constrained case that $\kappa(\mathcal{G})$ plays in the unconstrained case, and in fact reduces to $\kappa(\mathcal{G})$ if there are no constraints nearby, the following uniformity condition must be imposed in addition to Conditions 8.4 and 8.5.

CONDITION 8.8. *There exists $c_{\min} > 0$, independent of k, such that for all k*

$$c_{8.7}(\mathcal{G}_k) > c_{\min},$$

where $c_{8.7}(\mathcal{G})$ is the quantity from Proposition 8.7.

As in the unconstrained case, this ensures that at least one search direction lies safely within 90° of $-\nabla f(x)$.

This uniformity condition is automatically satisfied if there is only a finite number of distinct sets of search directions $\mathcal{G}_k$ allowed, as in the lattice-based GSS method in [168]. Otherwise, Condition 8.8 must be enforced extrinsically in the construction of the search directions. The satisfaction of Condition 8.8 is only a concern if some of the $K^\circ(x,\varepsilon)$ contain a subspace. In such cases, the goodness of the search directions needs to be ensured, as in the unconstrained case.

The other point necessary for the convergence analysis is that it is possible to take suitably long steps along the descent direction in $\mathcal{D}_k$ promised by Proposition 8.7 and still remain feasible. The following result ensures that it is possible to take reasonably long steps along directions in $K^\circ(x_k,\varepsilon)$ and remain feasible.

PROPOSITION 8.9. *If $x \in \Omega$, and $w \in K^\circ(x,\varepsilon)$ satisfies $\| w \| \leq \varepsilon$, then $x+w \in \Omega$.*

The proof follows from the definitions of $K(x,\varepsilon)$ and $K^\circ(x,\varepsilon)$.

8.2.5. The Situation at Unsuccessful Iterations. At unsuccessful iterations of Algorithms 8.1 and 8.2, there are bounds on the measure of stationarity $\chi(x_k)$. These bounds are linearly constrained versions of Theorem 3.3. As in the discussion of unsuccessful iterations in the unconstrained case, we make some assumptions to simplify the statements of the results that follow. In particular, the Lipschitz continuity of $\nabla f(x)$ can be replaced by uniform continuity.

For Algorithm 8.1, requiring Condition 8.4 for the set of search directions yields the following result.

THEOREM 8.10. *Suppose ∇f is Lipschitz continuous with constant M. Furthermore, suppose the iterates $\{x_k\}$ generated by Algorithm* 8.1 *lie in a bounded set, and that $\| \nabla f(x) \| \leq \gamma$ on this set. Then there exists $c_{8.10}$, independent of k, but depending on $c_{8.7}(\mathcal{G}_k)$, such that if k is an unsuccessful iteration of Algorithm* 8.1 *for linearly constrained optimization, then*

$$\chi(x_k) \leq c_{8.10} \left[M\, \Delta_k\, \beta_{\max} + \frac{\rho(\Delta_k)}{\Delta_k \beta_{\min}} \right].$$

An analogous bound at unsuccessful iterations for Algorithm 8.2 depends on whether Condition 8.4 or Condition 8.5 is imposed on the set of search directions.

THEOREM 8.11. *Suppose ∇f is Lipschitz continuous with constant M. Furthermore, suppose the iterates $\{x_k\}$ generated by Algorithm* 8.2 *lie in a bounded set, and that $\| \nabla f(x) \| \leq \gamma$ on this set. Then there exist $c_{8.11}$ and $r_{8.11}$ independent of k, but depending on $c_{8.7}(\mathcal{G}_k)$, such that if k is an unsuccessful iteration of Algorithm* 8.2 *for linearly constrained optimization, then*

$$\chi(x_k) \leq c_{8.11} \left[M\, \Delta_k\, \beta_{\max} + \frac{\rho(\Delta_k)}{\Delta_k \beta_{\min}} \right]$$

if the search directions satisfy Condition 8.4, *and*

$$\chi(x_k) \leq c_{8.11} \left[M\, \Delta_k\, \beta_{\max} + \frac{\rho(\Delta_k)}{\Delta_k \beta_{\min}} + r_{8.11} \frac{\gamma}{\beta_{\min}} \varepsilon_k \right]$$

if the search directions satisfy Condition 8.5.

8.2.6. Global Convergence Results. The bounds for χ at unsuccessful iterations can be used to establish global first-order convergence results for Algorithms 8.1 and 8.2. The results given here are a synthesis of a variety of results from [167, 168, 173, 175].

Theorems 8.10 and 8.11 make clear that if the search directions satisfy Condition 8.4, the crux of the convergence analysis lies in showing, as in the unconstrained case, that either

$$\liminf_{\substack{k\to+\infty \\ k\in\mathcal{U}}} \Delta_k = 0 \quad \text{or} \quad \lim_{\substack{k\to+\infty \\ k\in\mathcal{U}}} \Delta_k = 0.$$

However, such a result follows from the arguments already made in sections 3.7.1 and 3.7.2.

This leads to the following convergence result for Algorithm 8.1. Suppose f is continuously differentiable on $\mathbb{R}^n$, and $\nabla f(x)$ is Lipschitz continuous on $\mathbb{R}^n$ with constant M. If $\rho \equiv 0$, suppose in addition that $\mathcal{L}_f(x_0) \cap \Omega$ is bounded, where $\mathcal{L}_f(x_0)$ is the level set $\{\ x \mid f(x) \le f(x_0)\ \}$. Otherwise, assume only that $\{x_k\}$ remains in a bounded set. Furthermore, assume that either

1. the sufficient decrease condition (3.9) holds, or
2. the entries of A are rational, and the conditions (3.14), (3.15), (3.17), (3.21) hold to produce a rational lattice.

Then Algorithm 8.1 produces iterates $\{x_k\}$ for which

$$\liminf_{k\to+\infty} \chi(x_k) = 0\,.$$

That is, the sequence of iterates has a limit point that is a KKT point of (8.7).

The additional assumption that the matrix of constraints A is rational arises in connection with linearly constrained GSS algorithms that enforce only simple decrease. As discussed in section 3.7.2, these rely on a lattice structure to ensure global convergence. The matrix of constraints A is assumed to be rational to ensure that it is possible to construct search directions for $\mathcal{D}_k$ that lead to the desired lattice structure. See [168] for a further discussion of this point.

One can ensure that $\lim_{k\to+\infty} \chi(x_k) = 0$ by enforcing a feasible point version of Assumption 3.12 from the unconstrained case (see [168]).

In Algorithm 8.2, the sufficient decrease condition (3.9) is enforced. In [175] the authors show that

$$\lim_{k\to+\infty} \Delta_k = 0.$$

Moreover, if the search directions satisfy Condition 8.5, they show that there is a subsequence K of iterations for which

$$\lim_{k\in K} \varepsilon_k = 0.$$

With this additional information, Theorem 8.11 leads to the following result. Let f be continuously differentiable on $\mathbb{R}^n$ and $\nabla f(x)$ Lipschitz continuous on $\mathbb{R}^n$ with constant M. Suppose the iterates produced by Algorithm 8.2 remain in a bounded set. Then Algorithm 8.2 produces iterates $\{x_k\}$ for which

$$\liminf_{k\to+\infty} \chi(x_k) = 0\,.$$

If the search directions satisfy Condition 8.4, this result can be extended, again by enforcing a feasible point version of Assumption 3.12, to show that

$$\lim_{k \to +\infty} \chi(x_k) = 0.$$

That is, all the limit points of the sequence of iterates are KKT points.

8.2.7. Stopping Criteria. The results in section 8.2.5 suggest practical stopping criteria for GSS algorithms for linearly constrained problems. The situation is much the same as that discussed in section 3.10 for unconstrained problems.

Theorem 8.10 shows that at unsuccessful iterations, the continuous measure of stationarity χ will tend to decrease as Δ_k is decreased. Consequently, it is reasonable to terminate the algorithm when Δ_k falls below some tolerance. Theorem 8.11 tells us something similar, though if the search directions are only chosen to satisfy Condition 8.5, then not only Δ_k but also ε_k should be required to be sufficiently small before termination.

8.2.8. Constructing the Search Directions. If the generators of $K(x_k, \varepsilon)$ are linearly independent, then there are several ways in which to compute the generators of the polar cone $K^\circ(x_k, \varepsilon)$, which give the search directions needed for a GSS algorithm. For a further discussion of how to construct the search directions (or, equivalently, generators of a polyhedral cone), see [168, 175, 178, 210].

A simple construction is the following, taken from [168]. Suppose that for some $\varepsilon^* > 0$, the cone $K(x, \varepsilon^*)$ has a linearly independent set of generators (i.e., the constraints in (8.8)). Let V denote the matrix whose columns are these generators. If there are r generators, V is an $n \times r$ matrix. Let N be a positive basis for the null space of V^T. Then one can show that for any ε, $0 \leq \varepsilon \leq \varepsilon^*$, a set of generators of $K^\circ(x, \varepsilon)$ can be found among the columns of N, $V(V^TV)^{-1}$, and $-V(V^TV)^{-1}$.

This gives a simple way to compute generators for all the cones $K(x, \varepsilon)$ for all $0 \leq \varepsilon \leq \varepsilon^*$. Moreover, this construction shows that there need be no more than $2n$ generators in any given set of search directions $\mathcal{D}_k$, so long as the cone $K(x, \varepsilon^*)$ has a linearly independent set of generators. Thus, the sets of search directions for linearly constrained optimization need not contain a significantly larger number of search directions than those for unconstrained optimization. Generators can also be constructed using the QR factorization [175, 178].

If the generators of $K(x_k, \varepsilon)$ are not linearly independent, matters are much more complicated. In this case, the only vertex in $K^\circ(x_k, \varepsilon)$ (the origin) may be a *degenerate* vertex, where the notion of a degenerate vertex is that found in linear programming. The apex of the pyramid in Figure 8.7 is an example. The generators of $K^\circ(x_k, \varepsilon)$ correspond to edges of the cone, and the edges are determined by intersections of sets of $n - 1$ faces (hyperplanes corresponding to constraints in the original optimization problem). However, because more than n faces meet at the origin, not every set of $n - 1$ faces intersects in an edge of the cone and determines a generator. In this situation, there seems to be little recourse other than enumeration of all possible edges (see, for instance, [14] for a discussion of this problem).

8.3. General Nonlinear Constraints. We return to the general nonlinear program (7.1). The question of treating general nonlinear constraints with direct search optimization algorithms has a long history, and many different ideas have been proposed. There are difficulties, however, in extending direct search methods to general nonlinear programs.

8.3.1. Feasible Direction Methods That Use Constraint Derivatives. If the derivatives of the constraints $c(x)$ in (7.1) are available, or if estimation of these derivatives is deemed appropriate, then there are a number of proposed modifications of direct search methods along the lines of feasible directions algorithms. These methods involve the explicit use of derivatives of the constraints to compute feasible directions at the boundary of the feasible region.

Klingman and Himmelblau [157] give an algorithm with a simple construction of a single feasible direction that serves as the search direction. The method of Glass and Cooper [122] is more sophisticated and computes a new search direction by solving a linear programming problem involving a linear approximation of the objective and constraints, just as one would in a derivative-based feasible directions algorithm.

More recently, Lucidi, Sciandrone, and Tseng [175] presented a feasible directions, feasible iterates GSS approach for which one is guaranteed KKT point convergence. Algorithm 8.2 in section 8 is a specialization of their approach to the case of linear constraints.

The assumption underlying this approach is that the nonlinear constraints and their gradients can be computed relatively cheaply and accurately. This is the situation, for instance, in some shape optimization problems, in which the optimization variables x describe the shape of some object whose physical characteristics are being optimized. The constraints might be geometrical in nature, and given by fairly simple algebraic functions of x. The objective, on the other hand, might involve the adaptive solution of a differential equation, giving rise to an objective like that in Figure 1.2.

In the approach presented in [175], the feasible region near x_k is approximated by the linearization of the nearby constraints, as in traditional feasible directions methods that rely on derivatives. The important search directions are the generators of the cones normal to the cones generated by the nearby outward-pointing normals, as in the linearly constrained approach just discussed.

This approach employs a sufficient decrease condition of the form $\rho(t) = o(t)$. The amount of feasible descent required is computed by conducting curvilinear searches along the curves defined by the projections of the search directions onto the feasible set. These projections onto the feasible set guarantee feasibility of the iterates. Since the constraints and their derivatives are assumed to be relatively inexpensive to compute, these projections may be assumed to be relatively inexpensive to compute as well.

In the class of algorithms described in [175], steps are not allowed to violate feasibility with respect to the linearizations of the locally active constraints. If one cannot take a step along a search direction as far as dictated by the step-length control parameter, one simply goes as far as possible, as in Algorithm 8.2. This means that if x_k is near the boundary, some of the steps may be quite short. However, the step-length control parameter is not reduced, so these short steps are only in the search directions that intersect the boundary.

It is also possible to choose the search directions according to Condition 8.5, if desired, so one only works with the generators of a single cone $K^\circ(x_k, \varepsilon_k)$ at each iteration, rather than generators for the entire family $K^\circ(x_k, \varepsilon)$, $0 \le \varepsilon \le \varepsilon^*$. This idea was illustrated for the linearly constrained case in Algorithm 8.2. At unsuccessful iterations, ε_k is reduced as well as the step-length control parameters.

Under very mild assumptions on the nonlinear constraints $c(x)$, and the assumption that the sequence of iterates remains in a compact set, the following convergence results can be obtained [175]. If the search directions are chosen according to Condi-

tion 8.4, then all the limit points of the sequence of iterates are KKT points. If the search directions are chosen according to Condition 8.5, then at least one limit point of the sequence of iterates is a KKT point.

8.3.2. An Augmented Lagrangian Approach. Another approach to using GSS methods for nonlinearly constrained problems is the augmented Lagrangian approach by Lewis and Torczon in [169]. Augmented Lagrangian methods were first proposed in the late 1960s by Hestenes [135] and Powell [215]. The GSS variant is based on the augmented Lagrangian method of Conn, Gould, and Toint [68]. The latter algorithm is modified by using a GSS method for the bound constrained minimization of the augmented Lagrangian.

Assume the problem has been stated as an equality constrained problem, possibly after the introduction of slack variables in inequalities:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & c(x) = 0 \\ & \ell \le x \le u. \end{array} \tag{8.10}$$

The augmented Lagrangian in [68] is

$$\Phi(x;\lambda,\sigma,\mu) = f(x) + \sum_{i=1}^{m} \lambda^{(i)} c^{(i)}(x) + \frac{1}{2\mu} \sum_{i=1}^{m} \sigma^{(i)} c^{(i)}(x)^2.$$

The vector $\lambda^{(i)} = (\lambda_1^{(i)}, \ldots, \lambda_m^{(i)})^T$ is the Lagrange multiplier estimate for the equality constraints, μ is the penalty parameter, and the components $\sigma^{(i)}$ of the vector σ are positive weights. The equality constraints of (8.10) are incorporated in the augmented Lagrangian Φ while the simple bounds are left explicit. For a particular choice of multiplier estimate λ_k, penalty parameter μ_k, and scaling σ_k, define

$$\Phi_k(x) = \Phi(x;\lambda_k,\sigma_k,\mu_k).$$

At iteration k of the augmented Lagrangian algorithm described in [68], one approximately solves the subproblem

$$\begin{array}{ll} \text{minimize} & \Phi_k(x) \\ \text{subject to} & \ell \le x \le u. \end{array} \tag{8.11}$$

The degree to which this subproblem must be solved is given by

$$\| P_\Omega \left(x_k - \nabla \Phi_k(x_k) \right) \| \le \omega_k, \tag{8.12}$$

where P_Ω denotes the projection onto $\Omega = \{x \,|\, \ell \le x \le u\}$ and ω_k is updated at each iteration k.

The feature of interest in the GSS adaptation of this algorithm is the way in which the bound constrained subproblem (8.11) is approximately solved using a bound constrained GSS method. Since GSS methods do not have recourse to derivatives or explicit approximations thereof, the stopping criterion (8.12) is not appropriate. However, one can show that

$$\| P_\Omega \left(x_k - \nabla \Phi_k(x_k) \right) \| \le C \Delta_k,$$

a result similar to Theorem 8.3. This correlation between the step-length control parameter and a measure of stationarity makes it possible to replace (8.12) with a

stopping criterion based on the size of the step-length control parameter Δ. In effect, the algorithm proceeds by successive, inexact minimization of the augmented Lagrangian via a GSS method, but without knowing exactly how inexact the minimization is.

In the GSS version of the augmented Lagrangian algorithm, the Hestenes–Powell multiplier update formula is used since it does not require derivatives of the objective or constraints. The GSS augmented Lagrangian variant in [169] exhibits all of the first-order convergence properties of the original algorithm of Conn, Gould, and Toint.

If (8.10) is the result of introducing slacks to an inequality constrained problem, then further simplifications are possible to eliminate the slacks from the solution of (8.11). Decreasing the dimension of the problem improves the efficacy of direct search, as discussed previously.

8.3.3. Other Approaches. We conclude with a brief survey of other ideas that have been proposed for adapting direct search algorithms to nonlinear programs. While these approaches sometimes succeed in practice, there remains no guarantee that the search will approach a KKT point for the problem (7.1).

Inexact Penalization. A popular approach to constrained optimization has been to modify the objective so that it takes on very large values near the boundary of the feasible region and at all infeasible points. Rosenbrock [227], for example, proposed a form of inexact penalization as a means of treating constraints in connection with his rotating directions method. (A little-appreciated fact: Rosenbrock introduced his celebrated "banana" function in connection with direct search [227]. Rosenbrock's direct search method is tailored to deal with objectives whose graphs have long, narrow, turning valleys, such as that exhibited by the banana function.) Rosenbrock redefined the objective near the boundary of the feasible region in a way that would tend to keep the iterates feasible.

Similar ideas for modifying the objective in the case of bound constraints are discussed by Spendley, Hext, and Himsworth [245] and Nelder and Mead [194] in connection with the simplex-based methods. In these approaches the objective is given a suitably large value at all infeasible points. In the case of bound constraints, Spendley [244] also discusses the simple expedient of setting to the corresponding bound any variable that would otherwise become infeasible when applying the simplex algorithm of Nelder and Mead.

Exact Penalization. More systematic approaches to penalization also have appeared. The treatment of inequality constraints via exact, nonsmooth penalization (though not by that name) appears as early as the work of Hooke and Jeeves [139]. They suggested minimizing the nonsmooth penalty function

$$F(x) = f(x) + \sum_{i=1}^{m} w^{(i)} \max(0, c^{(i)}(x)),$$

where the weights $w^{(i)} > 0$ are chosen to be sufficiently large.

Weisman's `MINIMAL` algorithm [137] applies the pattern search algorithm of Hooke and Jeeves to a nonsmooth quadratic penalty function and incorporates an element of random search. More recently, Kearsley and Glowinski [124, 151] applied direct search methods with exact, nonsmooth penalization to equality constrained problems arising in control.

The potential difficulties with these approaches are those associated with nonsmooth optimization. While the direct search iterations always will be defined, there

remains the possibility that the algorithm will fail to converge to a stationary point, as discussed in section 6.

Barrier Methods. Davies and Swann [84], in connection with applying the pattern search method of Hooke and Jeeves to constrained optimization, recommend the use of the reciprocal barrier method of Carroll [54, 106]. One could also use a logarithmic barrier function.

Other Alternatives. There are number of direct search approaches to constrained optimization that do not neatly fit in the previous categories. These may be viewed as ad hoc methods insofar as there exist no guarantees of global convergence to KKT points. However, as already discussed in connection with direct search methods, good heuristics can often be placed on a rigorous footing. Certainly, they may prove useful in practice.

One direct search method for constrained optimization from the 1960s that has remained popular in application is M. J. Box's Complex method [39]. The Complex method was originally developed to address difficulties encountered with Rosenbrock's method. In this algorithm, the objective is sampled at a broader set of points than in the simplex-based methods to try to avoid premature termination. There is also an element of random search involved.

The `ACSIM` algorithm of Dixon [95] is a hybrid algorithm. It combines ideas from the simplex method of Nelder and Mead and the Complex method with elements of hem-stitching and quadratic modeling to accelerate convergence.

The flexible tolerance method of Paviani and Himmelblau [137, 207] takes the perspective of constrained optimization as a bilevel problem of improving feasibility and then optimality. The flexible tolerance method is based on the method of Nelder and Mead. It alternately attempts to reduce the objective and constraint violation, depending on the extent to which the iterates are infeasible. A strategy similar in spirit is used in Audet and Dennis [12], where a filter method inspired by that of Fletcher and Leyffer [108] is applied.

9. Summary. For the family of generating set direct search methods discussed in this paper, old worries about a lack of a firm mathematical foundation are no longer valid. In the unconstrained case, approaches such as compass search, evolutionary operation (EVOP) [35], the pattern search algorithm of Hooke and Jeeves [139], and multidirectional search [256] can all be assured of global convergence to a stationary point when the objective function is smooth.

Elements of the convergence analysis provide insights about the observed behavior of these methods:

- the geometry of the search directions (section 3.4) together with the globalization techniques (section 3.7) prevents bad steps in these algorithms, and explains their reliability;
- the cosine measure (3.10) and its dependence on the dimension of the problem help explain the observation that performance deteriorates for problems with large numbers of variables;
- the connection between a measure of stationarity and the step-length control parameter (section 3.6) offers a rationale for the standard termination criterion; and
- the local analysis (section 3.11) sheds light on why asymptotic convergence may be slow.

Understanding the elements needed to ensure convergence to KKT points has been crucial in devising reliable direct search methods for constrained optimization,

where the heuristics proposed in the past prove inadequate. When pursuing a feasible iterates approach to problems with constraints, the search directions must generate cones that reflect the geometry of nearby constraints. For nonlinear constraints, an approach that solves (approximately) a sequence of bound constrained or linearly constrained subproblems can be built on the relationship between an appropriately chosen measure of stationarity and the step-length control parameter.

Although derivatives do not appear explicitly, the "crypto-gradient" nature of generating set direct search methods emerges in analyzing their convergence. In fact, one can view these methods as lying at one end of a spectrum of methods (with Newton's method at the other end) that all require some smoothness to be assured that at least a subsequence of the iterates converges to a stationary point.

Work remains to explain why direct search methods so often experience practical success on noisy and nonsmooth problems, despite the existence of examples where they are known to fail. Gaps persist in the understanding of classic direct search methods that do not fit the GSS framework, such as the simplex algorithms of Nelder and Mead [194] and Spendley, Hext, and Himsworth [245].

There are opportunities for new, provably convergent direct search methods that are more efficient in practice because they adapt better to the local topography during the course of the search, as in [76]. In parallel, distributed, and heterogeneous computing environments, [115, 140, 161, 162] suggest ways to retain convergence properties while exploiting evolving computational paradigms. The issue of systematically improving efficiency on large-dimensional problems is open, as is the possibility of further hybrid algorithms that include elements of an explicit local model based on function values.

Finally, we reemphasize that, when direct search methods were originally developed, they were applied to problems for which much better alternatives are now available. Nevertheless, they are still a "method of first resort" for important and complex problems, especially in simulation-based optimization. Direct search methods can look forward to more years of practical success and theoretical analysis.

Acknowledgments. We thank Mike Heath for planting the seeds that led to this review. We thank Nick Trefethen for inviting us to submit a review for consideration by SIAM and for then patiently—and persistently—prodding us over the course of many years to deliver on our promise! We thank Randy LeVeque for shepherding us through the final rush of completion; he read multiple drafts of the manuscript (including the early, exceedingly rough versions), asked illuminating questions, and provided useful observations that did much to improve the presentation. We are particularly indebted to Margaret Wright, who not only read every draft of the manuscript we submitted and provided many useful suggestions, but who also expended a great deal of effort helping us to reorganize and edit the introduction and summary—their final form owes much to her efforts.

"Thanks, Grazie, and Gracias!" to Ian Coope, Stefano Lucidi, Juan Meza, Stephen Nash, Chris Price, Marco Sciandrone, and Michael Trosset for reading sundry versions of the manuscript and passing along corrections and suggestions. We are grateful to Anne Shepherd, who carefully read the entire manuscript, found mistakes others had missed, and made several suggestions for improving the presentation at key points. Finally, we thank the two anonymous referees who took great pains, under greatly restricted deadlines, to review the entire manuscript and pass along additional useful comments and corrections.

We also are grateful to Bas Franck for relating to us the hearing aid example in the introduction. Further, we are indebted to Jeff Borggaard for providing us with the "noisy" example in Figure 1.2.

Our thanks go out to the Computer Science Research Institute (CSRI) at Sandia National Laboratories for sponsoring many long-term visits between us, including a month-long visit during the summer of 2003 when we undertook this effort in earnest; we also are grateful to Karen Cardwell at Sandia for tracking down copies of the many older, and now difficult to find, references. We thank the IMA—the Institute for Mathematics and Its Applications—for the invitation to attend the Workshop on Optimization in Simulation-Based Models, January 9–16, 2003, which inspired some of the comments here and provided us a wonderful opportunity to work together on revisions to this paper during the week we were in residence.

We extend a general thanks to the many, many people who, knowing our interest in direct search methods, have asked us interesting (often tough!) questions, relayed their experience using direct search methods, and sent us copies of papers that we might not have discovered if left to our own devices. Our understanding of direct search is richer as a consequence.

REFERENCES

[1] E. R. Åberg and A. G. T. Gustavsson, *Design and evaluation of modified simplex methods*, Analytica Chimica Acta, 144 (1982), pp. 39–53.

[2] M. A. Abramson, *Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems*, Ph.D. thesis, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 2002.

[3] M. A. Abramson, C. Audet, and J. E. Dennis, Jr., *Generalized Pattern Searches with Derivative Information*, Tech. Rep. 02–10, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 2002.

[4] J. W. Akitt, *Function minimisation using the Nelder and Mead simplex method with limited arithmetic precision: The self regenerative simplex*, Comput. J., 20 (1977), pp. 84–85.

[5] P. Alberto, F. Nogueira, H. Rocha, and L. N. Vicente, *Pattern Search Methods for User-Provided Points: Application to Molecular Geometry Problems*, Tech. Rep. 00–20, Departamento de Matemática, Universidade de Coimbra, Portugal, 2000 (revised 2002).

[6] E. J. Anderson and M. C. Ferris, *A direct search algorithm for optimization with noisy function evaluations*, SIAM J. Optim., 11 (2001), pp. 837–857.

[7] H. L. Anderson, W. C. Davidon, M. G. Glicksman, and U. E. Kruse, *Scattering of positive pions by hydrogen at* 189 *MeV*, Phys. Rev., 100 (1955), pp. 279–287.

[8] Anonymous, *A new algorithm for optimization*, Math. Programming, 3 (1972), pp. 124–128.

[9] L. Armijo, *Minimization of functions having Lipschitz continuous first partial derivatives*, Pacific J. Math., 16 (1966), pp. 1–3.

[10] C. Audet, *Convergence Results for Pattern Search Algorithms Are Tight*, Tech. Rep. G–2002–56, Les Cahiers du GERAD, École Polytechnique de Montréal, Département de Mathématiques et de Génie Industriel, Montréal, QC, Canada, 2002.

[11] C. Audet and J. E. Dennis, Jr., *Pattern search algorithms for mixed variable programming*, SIAM J. Optim., 11 (2000), pp. 573–594.

[12] C. Audet and J. E. Dennis, Jr., *A Pattern Search Filter Method for Nonlinear Programming without Derivatives*, Tech. Rep. 00–09, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 2000.

[13] C. Audet and J. E. Dennis, Jr., *Analysis of generalized pattern searches*, SIAM J. Optim., 13 (2003), pp. 889–903.

[14] D. M. Avis and K. Fukuda, *A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra*, Discrete Comput. Geom., 8 (1992), pp. 295–313.

[15] M. Avriel, *Nonlinear Programming: Analysis and Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[16] J. W. Bandler and P. A. McDonald, *Optimization of microwave networks by razor search*, IEEE Trans. Microwave Theory Tech., 17 (1969), pp. 552–562.

[17] G. K. Barnes, *A Comparative Study of Nonlinear Optimization Codes*, Master's thesis, University of Texas at Austin, 1967.

[18] R. A. Barneson, N. F. Brannock, J. G. Moore, and C. Morris, *Picking optimization methods*, Chemical Engrg., 77 (1970), pp. 132–142.
[19] M. Bell and M. C. Pike, *Remark on algorithm* 178, Comm. ACM, 9 (1966), pp. 684–685.
[20] P. R. Benyon, *Remark AS R15. Function minimization using a simplex procedure*, Appl. Statist., 25 (1976), p. 97.
[21] G. Berman, *Minimization by successive approximation*, SIAM J. Numer. Anal., 3 (1966), pp. 123–133.
[22] G. Berman, *Lattice approximations to the minima of functions of several variables*, J. ACM, 16 (1969), pp. 286–294.
[23] D. Betteridge, A. P. Wade, and A. G. Howard, *Reflections on the modified simplex*—I, Talanta, 32 (1985), pp. 709–722.
[24] D. Betteridge, A. P. Wade, and A. G. Howard, *Reflections on the modified simplex*—II, Talanta, 32 (1985), pp. 723–734.
[25] C. Bischof, A. Carle, P. Khademi, and A. Mauer, *The ADIFOR* 2.0 *system for the automatic differentiation of Fortran* 77 *programs*, IEEE Comput. Sci. Engrg., 3 (1996), pp. 18–32.
[26] C. Bischof, A. Carle, P. Khademi, A. Mauer, and P. Hovland, *ADIFOR* 2.0 *User's Guide (Revision C)*, Tech. Rep. CRPC-95516-S, Center for Research on Parallel Computation, Rice University, Houston, TX, 1995.
[27] C. Bischof, L. Roh, and A. Mauer, *ADIC—An Extensible Automatic Differentiation Tool for ANSI-C*, Tech. Rep. ANL/CS-P626-1196, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1996.
[28] C. H. Bischof, A. Carle, G. Corliss, A. O. Griewank, and P. Hovland, *ADIFOR: Generating derivative codes from Fortran programs*, Sci. Programming, 1 (1992), pp. 11–29.
[29] P. T. Boggs and J. E. Dennis, Jr., *A stability analysis for perturbed nonlinear iterative methods*, Math. Comp., 30 (1976), pp. 199–215.
[30] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. B. Serafini, and V. Torczon, *Optimization using surrogate objectives on a helicopter test example*, in Optimal Design and Control, J. Borggaard, J. Burns, E. Cliff, and S. Schreck, eds., Progr. Systems Control Theory, Birkhäuser, Cambridge, MA, 1998, pp. 49–58.
[31] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset, *A rigorous framework for optimization of expensive functions by surrogates*, Structural Optimization, 17 (1999), pp. 1–13.
[32] J. Borggaard, D. Pelletier, and K. Vugrin, *On Sensitivity Analysis for Problems with Numerical Noise*, AIAA Paper 2002–5553; presented at the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA, 2002.
[33] D. M. Bortz and C. T. Kelley, *The simplex gradient and noisy optimization problems*, in Optimal Design and Control, J. Borggaard, J. Burns, E. Cliff, and S. Schreck, eds., Progr. Systems Control Theory, Birkhäuser, Cambridge, MA, 1998, pp. 77–90.
[34] G. E. P. Box, *The exploration and exploitation of response surfaces: Some general considerations and examples*, Biometrics, 10 (1954), pp. 16–60.
[35] G. E. P. Box, *Evolutionary operation: A method for increasing industrial productivity*, Appl. Statist., 6 (1957), pp. 81–101.
[36] G. E. P. Box and N. R. Draper, *Empirical Model-Building and Response Surfaces*, Wiley, New York, 1987.
[37] G. E. P. Box and N. R. Draper, *Evolutionary Operation: A Statistical Method for Process Improvement*, Wiley, New York, 1998.
[38] G. E. P. Box and K. B. Wilson, *On the experimental attainment of optimum conditions*, J. Roy. Statist. Soc. Ser. B, 13 (1951), pp. 1–45.
[39] M. J. Box, *A new method of constrained optimization and a comparison with other methods*, Comput. J., 8 (1965), pp. 42–52.
[40] M. J. Box, *A comparison of several current optimization methods, and the use of transformations in constrained problems*, Comput. J., 9 (1966), pp. 67–77.
[41] M. J. Box, D. Davies, and W. H. Swann, *Non-linear Optimization Techniques*, ICI Monograph 5, Oliver & Boyd, Edinburgh, UK, 1969.
[42] R. P. Brent, *Algorithms for Minimization without Derivatives*, Prentice-Hall, Englewood Cliffs, NJ, 1973; reissued by Dover Publications, Mineaola, NY, 2002.
[43] G. F. Brissey, R. B. Spencer, and C. L. Wilkins, *High-speed algorithm for simplex optimization calculations*, Analytical Chem., 51 (1979), pp. 2295–2297.
[44] K. W. Brodlie, *A new direction set method for unconstrained minimization without evaluating derivatives*, J. Inst. Math. Appl., 15 (1975), pp. 385–396.

[45] A. Brooke, D. Kendrick, and A. Meeraus, *GAMS: A Users' Guide*, The Scientific Press, Danvers, MA, 1988.
[46] S. H. Brooks, *A comparison of maximum-seeking methods*, Oper. Res., 7 (1959), pp. 430–457.
[47] S. H. Brooks and M. R. Mickey, *Optimum estimation of gradient direction in steepest ascent experiments*, Biometrics, 17 (1961), pp. 48–56.
[48] C. G. Broyden, *A class of methods for solving nonlinear simultaneous equations*, Math. Comp., 19 (1965), pp. 577–593.
[49] A. G. Buckley and H. Ma, *A Derivative-Free Algorithm for Parallel and Sequential Optimization*, tech. rep., Computer Science Department, University of Victoria, BC, Canada, 1994.
[50] J. Burkardt, M. Gunzburger, and J. Peterson, *Insensitive functionals, inconsistent gradients, spurious minima, and regularized functionals in flow optimization problems*, Int. J. Comput. Fluid Dyn., 16 (2002), pp. 171–185.
[51] K. W. C. Burton and G. Nickless, *Optimisation via simplex: Part* I. *Background, definitions and a simple application*, Chemometrics and Intelligent Laboratory Systems, 1 (1987), pp. 135–149.
[52] P. H. Calamai and J. J. Moré, *Projected gradient methods for linearly constrained problems*, Math. Programming, 39 (1987), pp. 93–116.
[53] F. M. Callier and P. L. Toint, *Recent results on the accelerating property of an algorithm for function minimization without calculating derivatives*, in Survey of Mathematical Programming, A. Prekopa, ed., Hungarian Academy of Sciences, Budapest, 1977, pp. 369–376.
[54] C. W. Carroll, *The created response surface technique for optimizing nonlinear, restrained systems*, Oper. Res., 9 (1961), pp. 169–185; with "Comments on the Preceding Paper" by Anthony V. Fiacco.
[55] M. R. Cave, *An improved simplex algorithm for dealing with boundary conditions*, Analytica Chimica Acta, 181 (1986), pp. 107–116.
[56] J. Céa, *Optimisation: Théorie et algorithmes*, Dunod, Paris, 1971.
[57] J. M. Chambers and J. E. Ertel, *Remark AS R11. A remark on algorithm AS* 47 *'Function minimization using a simplex procedure,'* Appl. Statist., 23 (1974), pp. 250–251.
[58] J. P. Chandler, *SIMPLEX—finds local minima of a function of several parameters (note)*, Behavioral Sci., 14 (1969), p. 82.
[59] J. P. Chandler, *STEPT: A family of routines for optimization and the fitting of data*, Tech. Rep. QCPE 307, Quantum Chemistry Program Exchange, Chemistry Department, Indiana University, Bloomington, IN, 1975.
[60] D. Chazan and W. Miranker, *Chaotic relaxation*, Linear Algebra Appl., 2 (1969), pp. 199–222.
[61] D. Chazan and W. L. Miranker, *A nongradient and parallel algorithm for unconstrained minimization*, SIAM J. Control, 8 (1970), pp. 207–217.
[62] D. H. Chen, Z. Saleem, and D. W. Grace, *A new simplex procedure for function minimization*, Internat. J. Modelling & Simulation, 6 (1986), pp. 81–85.
[63] K.-Z. Chen and Y.-P. Wang, *A New Direct Search Algorithm and Its Convergence*, Department of Applied Mathematics, Northwest Telecommunication Engineering Institute, Xi'an, China.
[64] F. L. Chernous'ko, *A local variation method for the numerical solution of variational problems*, U.S.S.R. Comput. Math. and Math. Phys., 5 (1965), pp. 234–242.
[65] T. D. Choi and C. T. Kelley, *Superlinear convergence and implicit filtering*, SIAM J. Optim., 10 (2000), pp. 1149–1162.
[66] F. H. Clarke, *Optimization and Nonsmooth Analysis*, Wiley, New York, 1983; reissued as Classics Appl. Math. 5, SIAM, Philadelphia, 1990.
[67] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Global convergence of a class of trust region algorithms for optimization with simple bounds*, SIAM J. Numer. Anal., 25 (1988), pp. 433–460.
[68] A. R. Conn, N. I. M. Gould, and P. L. Toint, *A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds*, SIAM J. Numer. Anal., 28 (1991), pp. 545–572.
[69] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust-Region Methods*, MPS/SIAM Ser. Optim. 1, SIAM, Philadelphia, 2000.
[70] A. R. Conn, K. Scheinberg, and P. L. Toint, *DFO (Derivative Free Optimization)*; software available at http://oss.software.ibm.com/developerworks/opensource/coin/.
[71] A. R. Conn, K. Scheinberg, and P. L. Toint, *On the convergence of derivative-free methods for unconstrained optimization*, in Approximation Theory and Optimization: Tributes

to M.J.D. Powell, M. D. Buhmann and A. Iserles, eds., Cambridge University Press, Cambridge, UK, 1997, pp. 83–108.

[72] A. R. Conn, K. Scheinberg, and P. L. Toint, *Recent progress in unconstrained nonlinear optimization without derivatives*, in Lectures on Mathematical Programming: ismp97, T. M. Liebling and D. de Werra, eds., Mathematical Programming 79, Series B, Elsevier, Amsterdam, 1997, pp. 397–414.

[73] A. R. Conn, K. Scheinberg, and P. L. Toint, *A derivative free optimization algorithm in practice*, in Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, 1998.

[74] A. R. Conn and P. L. Toint, *An algorithm using quadratic interpolation for unconstrained derivative free optimization*, in Nonlinear Optimization and Applications, G. Di Pillo and F. Giannessi, eds., Kluwer Academic/Plenum Publishers, New York, 1996, pp. 27–47.

[75] I. Coope and C. Price, *Frame-based methods for unconstrained optimization*, J. Optim. Theory Appl., 107 (2000), pp. 261–274.

[76] I. D. Coope and C. J. Price, *A direct search conjugate directions algorithm for unconstrained minimization*, ANZIAM J., 42 (2000), pp. C478–498.

[77] I. D. Coope and C. J. Price, *On the convergence of grid-based methods for unconstrained optimization*, SIAM J. Optim., 11 (2001), pp. 859–869.

[78] I. D. Coope and C. J. Price, *Positive bases in numerical optimization*, Comput. Optim. Appl., 21 (2002), pp. 169–175.

[79] R. J. Craig, *The Simplex Search in Statistical Optimization Problems*, Ph.D. thesis, University of Kentucky, Lexington, KY, 1980.

[80] R. J. Craig, J. W. Evans, and D. M. Allen, *The Simplex-Search in Non-linear Estimation*, Tech. Rep. 155, Department of Statistics, University of Kentucky, Lexington, KY, 1980.

[81] A. P. Dambrowski, *Variable-step simplex optimization procedures*, Tdch. Kibern., 1 (1970), p. 33.

[82] W. C. Davidon, *Variable Metric Method for Minimization*, Tech. Rep. 5990, Argonne National Laboratory, Argonne, IL, 1959.

[83] W. C. Davidon, *Variable metric method for minimization*, SIAM J. Optim., 1 (1991), pp. 1–17; with a belated preface for ANL 5990.

[84] D. Davies and W. H. Swann, *Review of constrained optimization*, in Optimization, R. Fletcher, ed., Academic Press, New York, 1969, pp. 187–202.

[85] O. L. Davies, *The Design and Analysis of Industrial Experiments*, Hafner Publishing, New York, 1954.

[86] C. Davis, *Theory of positive linear dependence*, Amer. J. Math., 76 (1954), pp. 733–746.

[87] R. De Leone, M. Gaudioso, and L. Grippo, *Stopping criteria for linesearch methods without derivatives*, Math. Programming, 30 (1984), pp. 285–300.

[88] R. De Vogelaere, *Remark on algorithm* 178, Comm. ACM, 11 (1968), p. 498.

[89] S. N. Deming and S. L. Morgan, *Simplex optimization of variables in analytical chemistry*, Analytical Chem., 45 (1973), pp. 278A–283A.

[90] S. N. Deming and L. R. Parker, Jr., *A review of simplex optimization in analytical chemistry*, CRC Critical Rev. Analytical Chem., 7 (1978), pp. 187–202.

[91] J. E. Dennis and V. Torczon, *Managing approximation models in optimization*, in Multidisciplinary Design Optimization: State of the Art, N. M. Alexandrov and M. Y. Hussaini, eds., SIAM, Philadelphia, 1997, pp. 330–347.

[92] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983; reissued as Classics Appl. Math. 16, SIAM, Philadelphia, 1996.

[93] J. E. Dennis, Jr. and V. Torczon, *Direct search methods on parallel machines*, SIAM J. Optim., 1 (1991), pp. 448–474.

[94] J. E. Dennis, Jr. and D. J. Woods, *Optimization on microcomputers: The Nelder–Mead simplex algorithm*, in New Computing Environments: Microcomputers in Large-Scale Computing, A. Wouk, ed., SIAM, Philadelphia, 1987, pp. 116–122.

[95] L. C. W. Dixon, *ACSIM—An accelerated constrained simplex technique*, Comput. Aided Design, 5 (1973), pp. 22–32.

[96] E. Dolan, A. Gurson, A. Shepherd, C. Siefert, V. Torczon, and A. Yates, *DirectSearch classes*; software available at http://www.cs.wm.edu/~va/software/DirectSearch.

[97] E. D. Dolan, *Pattern Search Behavior in Nonlinear Optimization*, Honors Thesis, Department of Computer Science, College of William & Mary, Williamsburg, VA, 1999.

[98] E. D. Dolan, R. M. Lewis, and V. J. Torczon, *On the local convergence properties of pattern search*, SIAM J. Optim., to appear.

[99] J. C. Dunn, *Global and asymptotic convergence rate estimates for a class of projected gradient processes*, SIAM J. Control Optim., 19 (1981), pp. 368–400.
[100] C. Elster and A. Neumaier, *A grid algorithm for bound constrained optimization of noisy functions*, IMA J. Numer. Anal., 15 (1995), pp. 585–608.
[101] S. V. Emel'yanov, S. K. Korovin, and A. S. Rykov, *Design principles and properties of locally optimal simplicial direct-search procedures*, Soviet Phys. Dokl., 24 (1979), pp. 532–534.
[102] F. E. Emery and M. O'Hagan, *Optimal design of matching networks for microwave transistor amplifiers*, IEEE Trans. Microwave Theory Tech., 14 (1966), pp. 696–698.
[103] J. W. Evans, *Computer augmentation of experimental designs to maximize $|X'X|$*, Technometrics, 21 (1979), pp. 321–330.
[104] P. G. Fajer, R. L. H. Bennett, C. F. Polnaszek, E. A. Fajer, and D. D. Thomas, *General method for multiparameter fitting of high-resolution EPR-spectra using a simplex algorithm*, J. Magnetic Resonance, 88 (1990), pp. 111–125.
[105] E. Fermi and N. Metropolis, *Los Alamos Unclassified Report LS–*1492, tech. rep., Los Alamos National Laboratory, Los Alamos, NM, 1952.
[106] A. V. Fiacco and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, Wiley, New York, 1968; reissued as Classics Appl. Math. 4, SIAM, Philadelphia, 1990.
[107] R. Fletcher, *Function minimization without evaluating derivatives—A review*, Comput. J., 8 (1965), pp. 33–41.
[108] R. Fletcher and S. Leyffer, *Nonlinear programming without a penalty function*, Math. Programming, Ser. A, 91 (2002), pp. 239–269.
[109] R. Fletcher, S. Leyffer, and P. L. Toint, *On the global convergence of a filter–SQP algorithm*, SIAM J. Optim., 13 (2002), pp. 44–59.
[110] R. Fletcher and M. J. D. Powell, *A rapidly convergent descent method for minimization*, Comput. J., 6 (1963), pp. 163–168.
[111] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, 2nd ed., Thomson/Brooks/Cole, Pacific Grove, CA, 2003.
[112] B. Franck, *Private communication*, Department of Experimental and Clinical Audiology, Academic Medical Centre, Amsterdam, 2002.
[113] P. Friedman and K. L. Pinder, *Optimization of a simulation model of a chemical plan*, Industrial & Engineering Chemistry Process Design and Development, 11 (1972), pp. 512–520.
[114] J. Gablonsky and C. T. Kelley, *A locally-biased form of the DIRECT algorithm*, J. Global Optim., 21 (2001), pp. 27–37.
[115] U. M. García-Palomares and J. F. Rodríguez, *New sequential and parallel derivative-free algorithms for unconstrained minimization*, SIAM J. Optim., 13 (2002), pp. 79–96.
[116] U. M. García-Palomares and J. F. Rodríguez, *Second-order information in the adaptive search exploration algorithm*, in Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, 2000, AIAA Paper 2000-4765.
[117] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.
[118] P. Gilmore, T. Choi, O. Eslinger, C. T. Kelley, H. A. Patrick, and J. Gablonsky, *IFFCO (Implicit Filtering For Constrained Optimization)*; software available at http://www4.ncsu.edu/~ctk/iffco.html.
[119] P. Gilmore and C. T. Kelley, *An implicit filtering algorithm for optimization of functions with many local minima*, SIAM J. Optim., 5 (1995), pp. 269–285.
[120] P. Gilmore, C. T. Kelly, C. T. Miller, and G. A. Williams, *Implicit filtering and optimal design problems*, in Optimal Design and Control, J. Borggaard, J. Burkhardt, M. Gunzberger, and J. Peterson, eds., Progr. Systems Control Theory 19, Birkhäuser, Cambridge, MA, 1995, pp. 159–176.
[121] T. Glad and A. Goldstein, *Optimization of functions whose values are subject to small errors*, BIT, 17 (1977), pp. 160–169.
[122] H. Glass and L. Cooper, *Sequential search: A method for solving constrained optimization problems*, J. ACM, 12 (1965), pp. 71–82.
[123] R. W. Glass and D. F. Bruley, *REFLEX method for empirical optimization*, Industrial and Engineering Chemistry Process Design and Development, 12 (1973), pp. 6–10.
[124] R. Glowinski and A. J. Kearsley, *On the simulation and control of some friction constrained motions*, SIAM J. Optim., 5 (1995), pp. 681–694.
[125] A. A. Goldstein, *Constructive Real Analysis*, Harper & Row, New York, 1967.

[126] A. O. Griewank, *On automatic differentiation*, in Mathematical Programming: Recent Developments and Applications, M. Iri and K. Tanabe, eds., Kluwer Academic, Dordrecht, The Netherlands, 1989, pp. 83–108.
[127] L. Grippo, F. Lampariello, and S. Lucidi, *Global convergence and stabilization of unconstrained minimization methods without derivatives*, J. Optim. Theory Appl., 56 (1988), pp. 385–406.
[128] J. A. Guin, *Discussion and correspondence—modification of the complex method of constrained optima*, Comput. J., 10 (1968), pp. 416–417.
[129] A. P. Gurson, *Simplex Search Behavior in Nonlinear Optimization*, Honors Thesis, Department of Computer Science, College of William & Mary, Williamsburg, VA, 2000.
[130] A. Gustavsson and J.-E. Sundkvist, *Design and optimization of modified simplex methods*, Analytica Chimica Acta, 167 (1985), pp. 1–10.
[131] W. E. Hart, *Evolutionary Pattern Search Algorithms*, Tech. Rep. SAND95–2293, Sandia National Laboratories, Albuquerque, NM, 1995.
[132] W. E. Hart, *A stationary point convergence theory for evolutionary algorithms*, in Foundations of Genetic Algorithms 4, Morgan Kaufmann, San Francisco, CA, 1996, pp. 325–342.
[133] W. E. Hart, *A generalized stationary point convergence theory for evolutionary algorithms*, in Proceedings of the International Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA, 1997, pp. 127–134.
[134] P. Hedlund and A. Gustavsson, *Design and evaluation of modified simplex methods having enhanced convergence ability*, Analytica Chimica Acta, 259 (1992), pp. 243–256.
[135] M. R. Hestenes, *Multiplier and gradient methods*, J. Optim. Theory Appl., 4 (1969), pp. 303–320.
[136] I. D. Hill, *Remark AS R28. A remark on algorithm AS* 47: *Function minimization using a simplex procedure*, Appl. Statist., 27 (1978), pp. 380–382.
[137] D. M. Himmelblau, *Applied Nonlinear Programming*, McGraw–Hill, New York, 1972.
[138] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms* I, Springer-Verlag, Berlin, 1993.
[139] R. Hooke and T. A. Jeeves, *Direct search solution of numerical and statistical problems*, J. ACM, 8 (1961), pp. 212–229.
[140] P. D. Hough, T. G. Kolda, and V. J. Torczon, *Asynchronous parallel pattern search for nonlinear optimization*, SIAM J. Sci. Comput., 23 (2001), pp. 134–156.
[141] P. D. Hough and J. C. Meza, *A class of trust-region methods for parallel optimization*, SIAM J. Optim., 13 (2002), pp. 264–282.
[142] E. C. Housos and O. Wing, *Pseudo-conjugate directions for the solution of the nonlinear unconstrained optimization problem on a parallel computer*, J. Optim. Theory Appl., 42 (1984).
[143] P. Hovland, *Automatic Differentiation and Its Role in Simulation-Based Optimization*, IMA Workshop, Minneapolis, MN, 2003.
[144] F. Y. Izakov, *Practical example of simplex-planning for seeking the optimal regimes of technological processes*, Industrial Laboratory, 37 (1971), pp. 420–422.
[145] F. James and M. Roos, *MINUIT—A system for function minimization and analysis of the parameter errors and correlations*, Comput. Phys. Comm., 10 (1975), pp. 343–367.
[146] M. G. Johnson, *Nonlinear optimization using the algorithm of Hooke and Jeeves*; software available in the Netlib Repository at http://www.netlib.org/opt/hooke.c, 1994.
[147] A. Jones, *Spiral—A new algorithm for non-linear parameter estimation using least squares*, Comput. J., 13 (1970), pp. 301–308.
[148] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, *Lipschitzian optimization without the Lipschitz constant*, J. Optim. Theory Appl., 79 (1993), pp. 157–181.
[149] A. F. Kaupe, Jr., *Algorithm* 178: *Direct search*, Comm. ACM, 6 (1963), pp. 313–314.
[150] A. J. Keane, *OPTIONS design exploration system*, Computational Engineering and Design Centre, School of Engineering Sciences, Mechanical Engineering, University of Southampton, Highfield, Southampton, UK; software available at http://www.soton.ac.uk/~ajk/options/welcome.html.
[151] A. J. Kearsley, *The Use of Optimization Techniques in the Solution of Partial Differential Equations from Science and Engineering*, Ph.D. thesis, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1996; available as Tech. Rep. 96–11, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1996.
[152] D. L. Keefer, *Simpat: Self-bounding direct search method for optimization*, Industrial and Engineering Chemistry Process Design and Development, 12 (1973), pp. 92–99.

[153] C. T. Kelley, *Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition*, SIAM J. Optim., 10 (1999), pp. 43–55.
[154] C. T. Kelley, *Iterative Methods for Optimization*, Frontiers Appl. Math. 18, SIAM, Philadelphia, 1999.
[155] J. Kiefer, *Optimum sequential search and approximation methods under minimum regularity assumptions*, J. Soc. Indust. Appl. Math., 5 (1957), pp. 105–136.
[156] P. G. King and S. N. Deming, *UNIPLEX: Single-factor optimization of response in the presence of error*, Analytical Chem., 46 (1974), pp. 1476–1481.
[157] W. R. Klingman and D. M. Himmelblau, *Nonlinear programming with the aid of a multiple-gradient summation technique*, J. ACM, 11 (1964), pp. 400–415.
[158] M. Kokkolaras, C. Audet, and J. E. Dennis, Jr., *Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system*, Optim. and Engrg., 2 (2001), pp. 5–29.
[159] T. G. Kolda, P. D. Hough, G. Gray, S. Brown, D. Dunlavy, and H. A. Patrick, *APPSPACK (Asynchronous Parallel Pattern Search package)*; software available at http://software.sandia.gov/appspack.
[160] T. G. Kolda, R. M. Lewis, and V. J. Torczon, *Stationarity results for generating set search for linearly constrained optimization*, in preparation, 2003.
[161] T. G. Kolda and V. J. Torczon, *On the Convergence of Asynchronous Parallel Pattern Search*, Tech. Rep. SAND2001–8696, Sandia National Laboratories, Livermore, CA, 2002; SIAM J. Optim., submitted.
[162] T. G. Kolda and V. J. Torczon, *Understanding asynchronous parallel pattern search*, in High Performance Algorithms and Software for Nonlinear Optimization, G. DiPillo and A. Murli, eds., Kluwer Academic, Dordrecht, The Netherlands, 2003, pp. 316–335.
[163] P. Krolak and L. Cooper, *An extension of Fibonaccian search to several variables*, Comm. ACM, 6 (1963), pp. 639–641.
[164] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, *Convergence properties of the Nelder–Mead simplex method in low dimensions*, SIAM J. Optim., 9 (1998), pp. 112–147.
[165] J. P. Lawrence III and F. P. Emad, *An adaptive randomized pattern search*, in Proceedings of 1972 IEEE Conference on Decision Control and 11th Symposium on Adaptive Processes, New Orleans, 1972, pp. 421–425.
[166] R. M. Lewis and V. Torczon, *Rank Ordering and Positive Bases in Pattern Search Algorithms*, Tech. Rep. 96–71, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, 1996.
[167] R. M. Lewis and V. Torczon, *Pattern search algorithms for bound constrained minimization*, SIAM J. Optim., 9 (1999), pp. 1082–1099.
[168] R. M. Lewis and V. Torczon, *Pattern search methods for linearly constrained minimization*, SIAM J. Optim., 10 (2000), pp. 917–941.
[169] R. M. Lewis and V. Torczon, *A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds*, SIAM J. Optim., 12 (2002), pp. 1075–1089.
[170] R. M. Lewis, V. Torczon, and M. W. Trosset, *Why pattern search works*, Optima, 59 (1998), pp. 1–7; also available as ICASE Tech. Rep. 98–57, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, 1998.
[171] R. M. Lewis, V. Torczon, and M. W. Trosset, *Direct search methods: Then and now*, J. Comput. Appl. Math., 124 (2000), pp. 191–207.
[172] S. Lucidi and M. Sciandrone, *Numerical results for unconstrained optimization without derivatives*, in Nonlinear Optimization and Applications, G. Di Pillo and F. Giannessi, eds., Kluwer Academic/Plenum Publishers, New York, 1996, pp. 261–270.
[173] S. Lucidi and M. Sciandrone, *A derivative-free algorithm for bound constrained optimization*, Comput. Optim. Appl., 21 (2002), pp. 119–142.
[174] S. Lucidi and M. Sciandrone, *On the global convergence of derivative-free methods for unconstrained optimization*, SIAM J. Optim., 13 (2002), pp. 97–116.
[175] S. Lucidi, M. Sciandrone, and P. Tseng, *Objective-derivative-free methods for constrained optimization*, Math. Program., 92 (2002), pp. 37–59.
[176] R. Luus and T. H. I. Jaakola, *Optimization by direct search and systematic reduction of the size of the search region*, AIChE J., 19 (1973), pp. 760–766.
[177] M. Marazzi and J. Nocedal, *Wedge trust region methods for derivative free optimization*, Math. Program., 91 (2002), pp. 289–305.

[178] J. H. May, *Linearly Constrained Nonlinear Programming: A Solution Method That Does Not Require Analytic Derivatives*, Ph.D. thesis, Yale University, New Haven, CT, 1974.
[179] K. I. M. McKinnon, *Convergence of the Nelder–Mead simplex method to a nonstationary point*, SIAM J. Optim., 9 (1998), pp. 148–158.
[180] J. C. Meza, R. S. Judson, T. R. Faulkner, and A. M. Treasurywala, *A comparison of a direct search method and a genetic algorithm for conformational searching*, J. Comput. Chem., 17 (1996), pp. 1142–1151.
[181] R. Mifflin, *A superlinearly convergent algorithm for minimization without evaluating derivatives*, Math. Programming, 9 (1975), pp. 100–117.
[182] R. Mifflin and J.-J. Strodiot, *A bracketing technique to ensure desirable convergence in univariate minimization*, Math. Programming, 43 (1989), pp. 117–130.
[183] R. Mifflin and J.-J. Strodiot, *A rapidly convergent five-point algorithm for univariate minimization*, Math. Programming, 62 (1993), pp. 299–319.
[184] R. A. Mitchell and J. L. Kaplan, *Nonlinear constrained optimization by a nonrandom complex method*, J. Res. Nat. Bur. Standards, 72C (1968), pp. 249–258.
[185] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Software, 7 (1981), pp. 17–41.
[186] J. J. Moreau, *Décomposition orthgonale d'un espace hilbertien selon deux cônes mutuellement polaires*, C. R. Acad. Sci. Paris, 255 (1962), pp. 238–240.
[187] E. Morgan, K. W. Burton, and G. Nickless, *Optimization using the modified simplex method*, Chemometrics and Intelligent Laboratory Systems, 7 (1990), pp. 209–222.
[188] E. Morgan, K. W. Burton, and G. Nickless, *Optimization using the super-modified simplex method*, Chemometrics and Intelligent Laboratory Systems, 8 (1990), pp. 97–107.
[189] S. L. Morgan and S. N. Deming, *Simplex optimization of analytical chemical methods*, Analytical Chem., 46 (1974), pp. 1170–1181.
[190] R. A. Mugele, *A program for optimal control of nonlinear processes*, IBM Systems J., 1 (1962), pp. 2–17.
[191] MultiSimplex^{TM} 2.0. Grabitech Solutions AB, Sundsvall, Sweden; software available at http://www.multisimplex.com, 2000.
[192] S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw–Hill, New York, 1996.
[193] L. Nazareth and P. Tseng, *Gilding the lily: A variant of the Nelder–Mead algorithm based on golden-section search*, Comput. Optim. Appl., 22 (2002), pp. 133–144.
[194] J. A. Nelder and R. Mead, *A simplex method for function minimization*, Comput. J., 7 (1965), pp. 308–313.
[195] L. S. Nelson, *A sequential simplex procedure for non-linear least-squares estimation and other function minimization problems*, in ASQC Technical Conference Transactions—Cleveland, 1972, American Society for Quality Control, Milwaukee, WI, pp. 107–117.
[196] J. Nocedal, *Theory of algorithms for unconstrained optimization*, Acta Numer., 1 (1992), pp. 199–242.
[197] J. Nocedal and S. Wright, *Numerical Optimization*, Springer Ser. Oper. Res., Springer-Verlag, New York, 1999.
[198] T. Öberg, *Importance of the first design matrix in experimental simplex optimization*, Chemometrics and Intelligent Laboratory Systems, 44 (1998), pp. 147–151.
[199] D. M. Olsson and L. S. Nelson, *The Nelder–Mead simplex procedure for function minimization*, Technometrics, 17 (1975), pp. 45–51.
[200] R. O'Neill, *Algorithm AS 47. Function minimization using a simplex procedure*, Appl. Statist., 20 (1971), pp. 338–345.
[201] R. O'Neill, *Corrigendum. Algorithm AS 47. Function minimization using a simplex procedure*, Appl. Statist., 23 (1974), p. 252.
[202] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970; reissued as Classics Appl. Math. 30, SIAM, Philadelphia, 2000.
[203] J. R. Palmer, *An improved procedure for orthogonalising the search vectors in Rosenbrock's and Swann's direct search optimisation methods*, Comput. J., 12 (1969), pp. 69–71.
[204] L.-R. Parker Jr., M. R. Cave, and R. M. Barnes, *Comparison of simplex algorithms*, Analytica Chimica Acta, 175 (1985), pp. 231–237.
[205] J. M. Parkinson and D. Hutchinson, *A consideration of non-gradient algorithms for the unconstrained optimization of functions of high dimensionality*, in Numerical Methods for Non-linear Optimization, F. A. Lootsma, ed., Academic Press, London, New York, 1972, pp. 99–113.

[206] J. M. Parkinson and D. Hutchinson, *An investigation into the efficiency of variants on the simplex method*, in Numerical Methods for Non-linear Optimization, F. A. Lootsma, ed., Academic Press, London, New York, 1972, pp. 115–135.
[207] D. Paviani and D. M. Himmelblau, *Constrained nonlinear optimization by heuristic programming*, Oper. Res., 17 (1969), pp. 872–882.
[208] G. Peckham, *A new method for minimising a sum of squares without calculating gradients*, Comput. J., 13 (1970), pp. 418–420.
[209] D. A. Phillips, *A preliminary investigation of function optimisation by a combination of methods*, Comput. J., 17 (1974), pp. 75–79.
[210] C. Pillers Dobler, *A matrix approach to finding a set of generators and finding the polar (dual) of a class of polyhedral cones*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 796–803.
[211] E. Polak, *Computational Methods in Optimization: A Unified Approach*, Academic Press, New York, 1971.
[212] E. Polak and M. Wetter, *Generalized Pattern Search Algorithms with Adaptive Precision Function Evaluations*, Department of Electrical Engineering, University of California at Berkeley, and Simulation Research Group, Building Technologies Department, Environmental Energy Technologies Division, Lawrence Berkeley National Laboratory, Berkeley, CA, 2001.
[213] M. J. D. Powell, *An efficient method for finding the minimum of a function of several variables without calculating derivatives*, Comput. J., 7 (1964), pp. 155–162.
[214] M. J. D. Powell, *A method for minimizing a sum of squares of non-linear functions without calculating derivatives*, Comput. J., 7 (1965), pp. 303–307.
[215] M. J. D. Powell, *A method for nonlinear constraints in minimization problems*, in Optimization, R. Fletcher, ed., Academic Press, New York, 1969, pp. 283–298.
[216] M. J. D. Powell, *A hybrid method for nonlinear equations*, in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon & Breach, London, 1970, pp. 87–114.
[217] M. J. D. Powell, *On search directions for minimization algorithms*, Math. Programming, 4 (1973), pp. 193–201.
[218] M. J. D. Powell, *Unconstrained minimization algorithms without computation of derivatives*, Boll. Un. Mat. Ital. (4), 9 (1974), pp. 60–69.
[219] M. J. D. Powell, *A view of unconstrained minimization algorithms that do not require derivatives*, ACM Trans. Math. Software, 1 (1975), pp. 97–107.
[220] M. J. D. Powell, *A direct search optimization method that models the objective and constraint functions by linear interpolation*, in Advances in Optimization and Numerical Analysis, Proceedings of the 6th Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico, S. Gomez and J.-P. Hennart, eds., Math. Appl. 275, Kluwer Academic, Dordrecht, The Netherlands, 1994, pp. 51–67.
[221] M. J. D. Powell, *Direct search algorithms for optimization calculations*, Acta Numer., 7 (1998), pp. 287–336.
[222] M. J. D. Powell, *On the Lagrange Functions of Quadratic Models That Are Defined by Interpolation*, Tech. Rep. DAMTP 2000/NA10, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK, 2000.
[223] C. J. Price and I. D. Coope, *Frame based ray search algorithms in unconstrained optimization*, J. Optim. Theory Appl., 116 (2003), pp. 259–377.
[224] C. J. Price, I. D. Coope, and D. Byatt, *A convergent variant of the Nelder Mead algorithm*, J. Optim. Theory Appl., 113 (2002), pp. 5–19.
[225] W. L. Price, *A controlled random search procedure for global optimisation*, Comput. J., 20 (1977), pp. 367–370.
[226] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.
[227] H. H. Rosenbrock, *An automatic method for finding the greatest or least value of a function*, Comput. J., 3 (1960), pp. 175–184.
[228] M. W. Routh, P. A. Swartz, and M. B. Denton, *Performance of the super modified simplex*, Analytical Chem., 49 (1977), pp. 1422–1428.
[229] T. H. Rowan, *Functional Stability Analysis of Numerical Algorithms*, Ph.D. thesis, University of Texas at Austin, 1990.
[230] P. B. Ryan, R. L. Barr, and H. D. Todd, *Simplex techniques for nonlinear optimization*, Analytical Chem., 52 (1980), pp. 1460–1467.
[231] A. S. Rykov, *Simplex direct search algorithms*, Automat. Remote Control, 41 (1980), pp. 784–793.
[232] A. S. Rykov, *Simplex methods of direct search*, Engrg. Cybernetics, 18 (1980), pp. 12–18.
[233] A. S. Rykov, *A combined penalty-function and projection method*, Soviet Phys. Dokl., 28 (1983), pp. 832–833.

[234] A. S. Rykov, *Simplex algorithms for unconstrained minimization*, Problems Control Inform. Theory, 12 (1983), pp. 195–208.
[235] R. W. H. Sargent, *Minimization without constraints*, in Optimization and Design, M. Avriel, M. J. Rijckaert, and D. J. Wilde, eds., Prentice-Hall, Englewood Cliffs, NJ, 1973, pp. 37–75.
[236] M. H. Shao, *Techniques and performance of the advanced regular simplex method*, Analytica Chimica Acta, 248 (1991), pp. 31–39.
[237] D. E. Shaw, R. W. M. Wedderburn, and A. Miller, *MINIM: A program for function minimization using the simplex method*; software available at StatLib: Data, Software and News from the Statistics Community, Applied Statistics algorithms, http://lib.stat.cmu.edu/apstat/47, 1991.
[238] H. Shekarforoush, M. Berthod, and J. Zerubia, *Direct Search Generalized Simplex Algorithm for Optimization Non-Linear Functions*, Tech. Rep. 2535, INRIA, France, 1995.
[239] C. M. Siefert, *Model-Assisted Pattern Search*, Honors Thesis, Department of Computer Science, College of William & Mary, Williamsburg, VA, 2000.
[240] C. M. Siefert and A. Yates, *MAPS (Model-Assisted Pattern Search)*; software available at http://www.cs.wm.edu/~va/software/maps.
[241] D. K. Smith, *A minimization solver for many parameters*, J. Quality Tech., 18 (1986), pp. 248–254.
[242] L. B. Smith, *Remark on algorithm* 178, Comm. ACM, 12 (1969), p. 638.
[243] H. A. Spang III, *A review of minimization techniques for nonlinear functions*, SIAM Rev., 4 (1962), pp. 343–365.
[244] W. Spendley, *Nonlinear least squares fitting using a modified simplex minimization method*, in Optimization, R. Fletcher, ed., Academic Press, New York, 1969, pp. 259–270.
[245] W. Spendley, G. R. Hext, and F. R. Himsworth, *Sequential application of simplex designs in optimisation and evolutionary operation*, Technometrics, 4 (1962), pp. 441–461.
[246] M. Strasser, *Übertrangung des Optimierungsverfahrens von Nelder und Mead auf restringierte Probleme*, Diploma thesis, Numerical Mathematics Group, Technical University of Darmstadt, Germany, 1994.
[247] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Wadsworth & Brooks/Cole, Pacific Grove, CA, 1989.
[248] N. Sugie, *An extension of Fibonaccian searching to multidimensional cases*, IEEE Trans. Automat. Control, 9 (1964), p. 105.
[249] W. H. Swann, *Report on the Development of a New Direct Search Method of Optimization*, Tech. Rep. Research Note 64/3, I.C.I. Central Instrument Lab, 1964.
[250] W. H. Swann, *Direct search methods*, in Numerical Methods for Unconstrained Optimization, W. Murray, ed., Academic Press, London, New York, 1972, pp. 13–28.
[251] W. H. Swann, *Constrained optimization by direct search*, in Numerical Methods for Constrained Optimization, P. E. Gill and W. Murray, eds., Academic Press, London, New York, 1974, pp. 191–217.
[252] S. W. Thomas, *Sequential Estimation Techniques for Quasi-Newton Methods*, Tech. Rep. TR75227, Department of Computer Science, Cornell University, Ithaca, NY, 1975.
[253] P. L. Toint and F. M. Callier, *On the accelerating property of an algorithm for function minimization without calculating derivatives*, J. Optim. Theory Appl., 23 (1977), pp. 531–547; correction by F. M. Callier and Ph. L. Toint, Optim. Theory Appl., 26 (1978), pp. 465–467.
[254] F. K. Tomlin and L. B. Smith, *Remark on algorithm* 178, Comm. ACM, 12 (1969), pp. 637–638.
[255] V. Torczon, *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*, Ph.D. thesis, Department of Mathematical Sciences, Rice University, Houston, TX, 1989; available as Tech. Rep. 90-07, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1990.
[256] V. Torczon, *On the convergence of the multidirectional search algorithm*, SIAM J. Optim., 1 (1991), pp. 123–145.
[257] V. Torczon, *On the convergence of pattern search algorithms*, SIAM J. Optim., 7 (1997), pp. 1–25.
[258] V. Torczon and M. W. Trosset, *From evolutionary operation to parallel direct search: Pattern search algorithms for numerical optimization*, Comput. Sci. Statist., 29 (1998), pp. 396–401.
[259] M. W. Trosset, *I know it when I see it: Toward a definition of direct search methods*, SIAG/OPT Views-and-News: A Forum for the SIAM Activity Group on Optimization, 9 (1997), pp. 7–10.

[260] M. W. Trosset, *On the Use of Direct Search Methods for Stochastic Optimization*, Tech. Rep. 00–20, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 2000.

[261] M. W. Trosset and V. Torczon, *Numerical Optimization Using Computer Experiments*, Tech. Rep. 97–38, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, 1997; revised 1999 and available as Tech. Rep. 97-2, Department of Computational & Applied Mathematics, Rice University, Houston, TX, 1997.

[262] P. Tseng, *Fortified-descent simplicial search method: A general approach*, SIAM J. Optim., 10 (1999), pp. 269–288.

[263] P. Tseng, *Convergence of block coordinate descent method for nondifferentiable minimization*, J. Optim. Theory Appl., 109 (2001), pp. 475–494.

[264] T. Umeda and A. Ichikawa, *A modified complex method for optimization*, Industrial and Engineering Chemistry Process Design and Development, 10 (1971), pp. 229–236.

[265] P. F. A. van der Wiel, R. Maassen, and G. Kateman, *The symmetry-controlled simplex optimization procedure*, Analytica Chimica Acta, 153 (1983), pp. 83–92.

[266] D. A. Walmsley, *The Simplex Method for Minimisation of a General Function*, Supplementary Report 686, Assessment Division, Transport Systems Department, Transport and Road Research Laboratory, Crowthorne, Berkshire, UK, 1981.

[267] F. H. Walters, L. R. Parker, Jr., S. L. Morgan, and S. N. Deming, *Sequential Simplex Optimization*, Chemometrics Series, CRC Press, Boca Raton, FL, 1991.

[268] D. Winfield, *Function minimization by interpolation in a data table*, J. Inst. Math. Appl., 12 (1973), pp. 339–347.

[269] P. Wolfe, *Convergence conditions for ascent methods*, SIAM Rev., 11 (1969), pp. 226–235.

[270] C. F. Wood, *Review of design optimization techniques*, IEEE Trans. Systems Sci. Cybernetics, 1 (1965), pp. 14–20.

[271] D. J. Woods, *An Interactive Approach for Solving Multi-Objective Optimization Problems*, Ph.D. thesis, Department of Mathematical Sciences, Rice University, Houston, TX, 1985; also available as TR85-05, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1985.

[272] M. H. Wright, *Direct search methods: Once scorned, now respectable*, in Numerical Analysis 1995 (Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis), D. F. Griffiths and G. A. Watson, eds., Pitman Res. Notes Math. Ser. 344, CRC Press, Boca Raton, FL, 1996, pp. 191–208.

[273] M. H. Wright, *What, If Anything, Is New in Optimization?*, Tech. Rep. 00–4–08, Computing Sciences Research Center, Bell Laboratories, Murray Hill, NJ, 2000.

[274] S. E. Wright, *A note on positively spanning sets*, Amer. Math. Monthly, 107 (2000), pp. 364–366.

[275] Y.-X. Yuan, *An example of non-convergence of trust region algorithms*, in Advances in Nonlinear Programming, Y.-X. Yuan, ed., Kluwer Academic, Dordrecht, The Netherlands, 1998, pp. 205–215.

[276] L. A. Yarbo and S. N. Deming, *Selection and preprocessing of factors for simplex optimization*, Analytica Chimica Acta, 73 (1974), pp. 391–398.

[277] W. Yu, *The convergence property of the simplex evolutionary techniques*, Scientia Sinica, Special Issue of Mathematics, 1 (1979), pp. 68–77.

[278] W. Yu, *Positive basis and a class of direct search techniques*, Scientia Sinica, Special Issue of Mathematics, 1 (1979), pp. 53–67.

[279] W. I. Zangwill, *Minimizing a function without calculating derivatives*, Comput. J., 10 (1967), pp. 293–296.

[280] E. H. Zarantonello, *Projections on convex sets in Hilbert space and spectral theory*, in Contributions to Nonlinear Functional Analysis, E. H. Zarantonello, ed., Academic Press, London, New York, 1971, pp. 237–424.

[281] S. K. Zavriev, *On the global optimization properties of finite-difference local descent algorithms*, J. Global Optim., 3 (1993), pp. 67–78.